VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

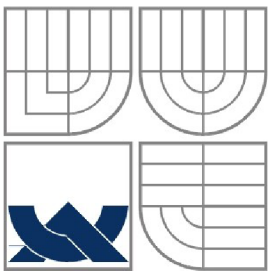
IRC ROBOT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

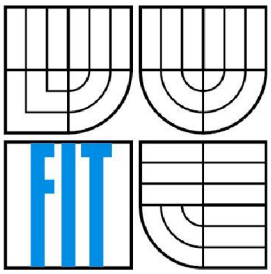
AUTOR PRÁCE
AUTHOR

MARTIN RAPAVÝ

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IRC ROBOT

IRC ROBOT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN RAPAVÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV RÁB

BRNO 2008

Zadání bakalářské práce/6494/2007/xrapav00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2007/2008

Zadání bakalářské práce

Řešitel: **Rapavý Martin**
Obor: Informační technologie
Téma: **IRC robot**
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s existujícími IRC roboty, IRC protokolem a funkcí IRC serverů
2. Navrhněte IRC robot pro IRC server, který bude splňovat následující požadavky: podpora IRC protokolu podle RFC 1459, správa uživatelů a jejich rolí, správa kanálů, administrativní interface
3. Implementujte navrženého IRC robota ve vhodném prostředí
4. Proveďte testování implementace a vyhodnoťte vhodnost implementace
5. Zhodnťte dosažené výsledky a diskutujte další možnost pokračování v projektu

Literatura:

- RFC 1459 (http://www.faqs.org/rfcs/rfc1459.html)

Při obhajobě semestrální části projektu je požadováno:

- Body 1. a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Ráb Jaroslav, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2007
Datum odevzdání: 14. května 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Martin Rapavý**
Id studenta: 78775
Bytem: Golianova 119, 949 11 Nitra
Narozen: 14. 05. 1986, Nitra
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: IRC robot
Vedoucí/školitel VŠKP: Ráb Jaroslav, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Táto bakalárska práca sa zaoberá analýzou aplikačného protokolu IRC, hlavne jeho klientskou časťou a možnosťami použitia. Práca podáva ucelený prehľad funkcií IRC robotov a zoznam ich súčasných implementácií. Posledné kapitoly rozoberajú návrh programu – IRC robota z pohľadu klientskej časti IRC protokolu ako aj z pohľadu možností užívateľskej konfigurácie. Na záver práca popisuje implementáciu navrhnutého programu.

Kľúčové slová

IRC, Internet Relay Chat, IRC robot, RFC 1459, RFC 2810, RFC 2812

Abstract

This bachelor's thesis is dedicated to application protocol IRC, mainly to its client subset and usage possibilties. The thesis summarizes functionality of IRC robots and list of their current implementations. Latter chapters describe new IRC robot program design, mainly concerning IRC client protocol and possibilties of user configuration. In the end, the thesis describes implementation of designed program.

Keywords

IRC, Internet Relay Chat, IRC robot, RFC 1459, RFC 2810, RFC 2812

Citácia

Martin Rapavý: IRC robot, bakalárska práca, Brno, FIT VUT v Brně, 2008

IRC robot

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Jaroslava Rába. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……
Martin Rapavý
30.4.2008

Poďakovanie

Ďakujem vedúcemu mojej bakalárskej práce Ing. Jaroslavovi Rábovi za poskytnuté rady a názory.

Obsah

1 Úvod

Medzi často využívané služby Internetu patrí elektronická komunikácia formou textových správ. Spôsobov, ktorými si užívatelia správy vymieňajú existuje niekoľko. Podľa potreby je možné vyžiť neinteraktívne prostriedky (e-mail, netnews) alebo interaktívne prostriedky (instant-messaging, chat) elektronickej komunikácie.

V súčasnej dobe prevládajú na poli interaktívnej elektronickej komunikácie protokoly typu „instatnt-messaging“, teda výmena krátkych textových správ medzi dvomi užívateľmi v reálnom čase. V úzadí zostávajú protokoly, ktoré vytvárajú možnosti konferencie viacerých užívateľov v reálnom čase. Medzi systémy, ktoré boli vyvinuté s dôrazom na poskytovanie kvalitnej viac užívateľskej konferencie v reálnom čase patria napr. SILC (*Secure Internet Live Conferencing*) a IRC (*Internet Relay Chat*).

Táto bakalárska práca sa zaoberá systémom IRC, protokolom, na ktorom je postavený a možnosťami, ktoré poskytuje.

Druhá kapitola poskytuje prehľad histórie a súčasného stavu protokolu IRC, popis jeho fungovania a architektúry. Nakoľko je tento protokol značne rozsiahly, sú v práci zaznamenané len aspekty, ktoré súvisia s komunikáciou IRC klienta a IRC servera.

Tretia kapitola zoznamuje čitateľa s možnosťami automatizácie určitých úkonov spojených s rolou tzv. „operátora“ IRC kanálu – s IRC robotmi. V úvode kapitoly je popísaná požadovaná funkcionalita. Ďalej nasleduje stručný prehľad existujúcich implementácií, ich výhody a nevýhody.

V štvrtej kapitole je popísaný návrh nového IRC robota, potrebných dátových štruktúr a operácií. Kapitola sa tiež venuje rozdeleniu návrhu programu do jednotlivých modulov a ich vzájomnému prepojeniu. Textový popis návrhu je pre zjednodušenie sprevádzaný obrázkami a diagramami.

Popis výberu implementačného prostredia, samotnej implementácie, problémov, ktoré bolo pri nej nutné riešiť a architektúry celého programu uvádza piata kapitola.

Záver práce zhodnocuje dosiahnuté výsledky a možnosti ďalšieho vývoja projektu.

2 Systém IRC

2.1 História

Začiatky vývoja protokolu IRC siahajú do roku 1988. Na finskej univerzite *Oulu* začal s jeho vývojom jeden zo zamestnancov – Jarkko Oikarinen. Jeho cieľom bolo nahradiť program MUT (*MultiUser Talk*), ktorý bol v tom čase používaný za účelom viac užívateľskej interaktívnej konferencie. Pri návrhu protokolu sa autor inšpiroval systémom *Bitnet Relay*, ktorý sa používal v americkej akademickej sieti BITNET.

Vývoj a práce na implementácií prvého IRC servera a IRC klienta pokračovali do roku 1993, kedy vznikol prvý pokus o štandardizáciu protokolu formou dokumentu RFC 1459 [1]. V tomto čase sa už objavovali nezhody medzi jednotlivými vývojármi ohľadom istých technických problémov súvisiacich s prevádzkou viacerých IRC serverov spojených do IRC siete. Vývoj IRC protokolu a IRC serverov sa následne uberal niekoľkými vetvami. Určitý pokus o nápravu tohto stavu nastal v roku 2000, kedy vyšli dokumenty RFC 2810 [2], RFC 2811, RFC 2812 [3], RFC 2813, ktoré upravovali pôvodný „štandard“ RFC 1459 a aktualizovali ho o nové techniky implementované v predchádzajúcich rokoch. Tieto zmeny však neboli všetkými, v tom čase existujúcimi, implementáciami IRC servera prijaté.

V súčasnosti existuje veľké množstvo bežiacich IRC serverov a IRC sietí, ktoré používajú rôzne upravený pôvodný IRC štandard. Medzi najrozsiahlešie 4 siete súčasnosti patria:

- IRCnet – sieť pozostávajúca z pôvodných európskych serverov,
- EFnet – sieť severoamerických serverov, ktorá sa po nezhodách odpojila od tých európskych,
- Undernet,
- DALnet.

V strednej Európe je medzi užívateľmi najpoužívanejšia sieť IRCnet, do ktorej je pripojený aj prvý IRC server vôbec – tolsun.oulu.fi.

Okrem spomínaných veľkých sietí vznikli aj niektoré menšie, zamerané na určitú problematiku:

- Freenode – používajú rôzne projekty na koordinovanie vývoja,
- Quakenet – používaný najmä hráčmi počítačových hier,
- SciFi – sieť zameraná na žáner vedecko-fantastickej literatúry a filmu.

2.2 Architektúra

Systém IRC je typickým predstaviteľom modelu aplikácie klient-server. IRC navyše ponúka možnosť z niekoľkých serverov vytvoriť IRC sieť, ktorá distribuovane poskytuje svoje služby užívateľom.

2.2.1 Užívatelia

Užívateľom IRC siete je každý klient, ktorý sa úspešne prihlási na jeden z jej serverov. Každý užívateľ je identifikovaný trojicou údajov:

- prezývka *(nickname)*,
- ident,
- doménové meno počítača, z ktorého sa klient pripája (*hostname*).

Tieto údaje sa zapisujú vo formáte:

```
nickname!ident@hostname
```

Tomuto formátu sa hovorí maska, pretože je možné pomocou zástupných znakov hviezdička (*) a otáznik (?) určiť množinu užívateľov, ktorí takejto maske vyhovujú. Hviezdička (*asterisk*) zastupuje ľubovoľný počet ľubovoľných znakov. Otáznik zastupuje práve jeden ľubovoľný znak.

Prvý z nich má možnosť si v rámci pravidiel zvoliť samotný užívateľ. Ident [4] je identifikátor klientovho TCP spojenia na IRC server, ktorý si server zistí z ident servera bežiaceho u klienta. V prípade, že klient nemá takúto službu na svojom počítači spustenú, alebo je neprístupná, zvolí si ident sám. Užívateľsky zvolený ident IRC server označí špeciálnym prefixom, aby bolo pre ostatných užívateľov zrejmé, že tento údaj môže byť nepravdivý. Doménové meno klientovho počítača IRC server zistí vyhľadaním reverzného DNS záznamu k IP adrese, z ktorej sa klient pripája. V prípade, že takýto záznam neexistuje, použije server klientovu IP adresu.

Ďalšou identifikáciou každého užívateľa je unikátny identifikátor, ktorý sa skladá z identifikácie IRC servera, ku ktorému je užívateľ pripojený a lokálne jednoznačného reťazca. Tento údaj obdrží klient pri pripájaní. Jeho funkcia je popísaná v nasledujúcom texte.

2.2.2 Servery

IRC servery predstavujú centrálne prvky, ku ktorým sa pripájajú IRC klienti a využívajú ich služieb. IRC servery majú možnosť spájať sa a vytvoriť tak IRC sieť. Servery smú vytvoriť len stromovú topológiu bez kružníc (*spanning tree*).

O doručovanie správ v rámci IRC siete sa starajú servery medzi sebou a pre klienta je táto činnosť transparentná. Preposielanie správ *(message relaying)* uskutočňujú servery na základe jednoznačného identifikátora priradeného každému klientovi pri pripájaní na server. Tento jednoznačný identifikátor začína číslom, ktoré identifikuje krajinu a server v nej. Číselné identifikácie

serverov sú prideľované podľa normy ISO3166. Lokálny IRC server následne pridelí užívateľovi reťazec jednoznačne ho identifikujúci v rámci daného servera. Tento reťazec spolu s číselnou identifikáciou servera, ku ktorému je klient pripojený, dáva užívateľovi unikátny identifikátor v rámci celej IRC siete. Unikátne identifikátory užívateľov (*user unique identification*) používajú servery pri preposielaní si správ medzi sebou. Týmto mechanizmom riešia napríklad problém, kedy adresát správy počas jej prenosu zmení prezývku. Pred doručením správ adresátovi je, podľa očakávania IRC klienta, preložený unikátny identifikátor späť na jeho prezývku.

Protokol IRC poskytuje užívateľom so zvláštnym postavením prostriedky na správu IRC serverov a spojení medzi nimi. Títo užívatelia sa nazývajú IRC operátori a udržujú bezproblémovú prevádzku siete.

2.2.3 Užívateľská komunikácia

Cieľom protokolu IRC je umožniť užívateľom výmenu textových správ. Podľa typu adresáta správy je možné rozdeliť komunikáciu medzi užívateľmi do nasledujúcich kategórií: užívateľ užívateľovi, užívateľ zoznamu užívateľov, užívateľ skupine užívateľov, užívateľ všetkým užívateľom.

2.2.3.1 Užívateľ užívateľovi

Ide o najjednoduchší spôsob komunikácie. Odosielateľ uvedie ako adresáta správy prezývku užívateľa, ktorému chce správu odoslať. Pokiaľ je to možné, IRC sieť správu doručí, v opačnom prípade sa odosielateľovi vráti hlásenie s popisom chyby, kvôli ktorej nebolo správu možné doručiť. Najčastejším dôvodom nedoručiteľnosti správy je jej neplatný alebo neexistujúci adresát.

2.2.3.2 Užívateľ zoznamu užívateľov

Typ komunikácie takmer totožný s predchádzajúcim. Rozdiel spočíva v tom, že správa nie je adresovaná jednému užívateľovi, ale zoznamu užívateľov oddelených medzerou.

2.2.3.3 Užívateľ skupine užívateľov

Tento spôsob komunikácie je pre IRC typický a je aj najviac používaný. Správa je v tomto prípade adresovaná pomenovanej skupine užívateľov, ktorá sa nazýva IRC kanál. Tieto kanály svojou povahou a funkciou pripomínajú IP multicastové skupiny. Užívatelia sa na ne môžu pripojiť a následne prijímať všetky správy, ktoré sú tomuto kanálu adresované, alebo sa z kanálu odpojiť. Po pripojení nového užívateľa na kanál mu je ako prvá správa zaslaná téma kanálu (*channel topic*) a zoznam pripojených užívateľov. Tému kanálu môže v závislosti na nastavení kanálu meniť buď každý alebo len operátor daného kanálu. IRC kanál vzniká implicitne pripojením prvého užívateľa a zaniká odpojením posledného užívateľa.

IRC kanály sa z hľadiska rozsahu pôsobnosti delia na dve skupiny: lokálne a globálne. Názvy globálnych kanálov začínajú symbolom „#“. Vytvorenie globálneho kanála je distribuované do celej IRC siete. Takýto kanál je v celej IRC sieti unikátny a užívateľ pripojený na ľubovoľný server siete sa menom tohto kanála odkáže vždy na ten istý kanál. Názvy lokálnych kanálov začínajú symbolom &. Platnosť existencie takého kanálu je obmedzená na lokálny IRC server, na ktorom bol vytvorený. Na každom serveri IRC siete teda môže existovať iný lokálny kanál s tým istým menom.

IRC kanálom je možné nastavovať atribúty, ktoré prispôsobujú vlastnosti kanálu užívateľským potrebám. Tieto atribúty sa v terminológií IRC nazývajú kanálové módy *(channel modes)*. Prístup k zmene módov majú len operátori daného kanálu. Prehľad všetkých módov a ich význam je uvedený v nasledujúcej tabuľke.

Názov módu	Skratka	Význam
operator	o	Modifikácia práva operátor kanálu
private	p	Súkromný kanál; v zozname všetkých kanálov je označený ako súkromný a nie je zobrazená jeho téma
secret	s	Tajný kanál; nie je zobrazený v zozname všetkých kanálov
invite-only	i	Kanál len na pozvanie; môžu sa pripojiť len užívatelia, ktorých pozval niektorý z operátorov kanálu
topic-protected	t	Kanál s chránenou témou; tému kanálu môžu meniť len operátori
no external messages	n	Na kanál môžu prispievať len užívatelia, ktorí sú naň pripojení
moderated	m	Moderovaný kanál; môžu prispievať len operátori a užívatelia s právom voice
limit	l	Kanál s obmedzenou kapacitou; na kanál sa môže pripojiť zadané maximum užívateľov
ban	b	Zákaz prístupu; na kanál sa nemôžu pripojiť užívatelia, ktorých maska sa zhoduje s niektorou z masiek zadaných v tzv. *ban-liste*
voice	v	Modifikácia práva voice
key	k	Nastavenie kľúča kanálu; na kanál sa môžu pripojiť len užívatelia, ktorí poznajú jeho kľúč
exeception	e	Modfikácia zoznamu masiek, ktoré sú vyňaté z *ban-listu*
invitation	I	Modifikácia zoznamu automaticky pozvaných užívateľov
server reop	R	Modifikácia zoznamu užívateľov, ktorým server pridelí právo operátora, ak sa kanál ocitne dlhší čas (typicky 90 min.) bez operátorov

Tabuľka 2.1: Prehľad kanálových módov

Na každom kanáli sa nachádzajú tri skupiny užívateľov: operátori, užívatelia s právom *voice* a normálni užívatelia.

Operátori starajú o administráciu kanála. Prvý užívateľ, ktorý príde na kanál (vytvorí kanál) sa automaticky stáva jeho operátorom. Tento užívateľ môže podľa uváženia pridať alebo odobrať ľubovoľnému užívateľovi kanála právo operátora. Rovnaké možnosti má každý ďalší operátor.

Užívatelia s právom *voice* majú možnosť prispievať na kanály, ktoré sú tzv. moderované. Na moderovaný kanál môžu prispievať aj operátori kanála. Toto je možné využiť pri rozhovoroch typu interview, kde sa úzka skupina užívateľov pýta alebo odpovedá a ostatní majú možnosť diskusiu len pasívne sledovať. Nastaviť kanál do módu moderovaného kanálu majú jeho operátori. Operátori majú možnosť nastavovať aj viacero ďalších módov kanálu, ktoré budú detailne popísané v ďalších kapitolách.

Normálni užívatelia majú právo posielať na kanál správy, ak momentálne nie je moderovaný a meniť tému, na ktorú sa diskutuje, ak to nie je atribútom „`t`“ zakázané.

2.2.3.4 Užívateľ všetkým užívateľom

Tento spôsob komunikácie je v IRC zahrnutý, aby mali IRC operátori prostriedok na informovanie všetkých užívateľov o prevádzkových informáciách, plánovaných odstávkach apod. Doručenie týchto správ je možné v prípade potreby obmedziť na určitú skupinu užívateľov a to podľa masky, ktorá na vhodných miestach (prezývka, ident, host) obsahuje niektorý zo zástupných znakov uvedených v podkapitole 2.2.1.

2.3 Protokol

IRC používa na komunikáciu medzi klientom a serverom ako aj medzi servermi navzájom textový protokol. Správy sa pri prenose kódujú do postupnosti jednobajtových znakov. Štandard nešpecifikuje znakovú sadu, ktorá sa má použiť, preto je vždy bezpečné použiť len US-ASCII. Pri použití znakov národných znakových sád je zodpovednosť za ich správnu interpretáciu na klientoch.

2.3.1 Syntax správ

Pri IRC komunikácií si klient so serverom, ako aj servery medzi sebou, vymieňajú IRC správy. Každý riadok komunikácie predstavuje jednu IRC správu. Správy sú navzájom oddelené dvojicou znakov *carriage return* a *line feed* (CR, LF). Je povolené použiť medzi správami aj viacero dvojíc znakov CR a LF, pričom prázdne správy, ktoré týmto spôsobom vzniknú sú bez varovania ignorované.

Správa pozostáva z troch častí:

- prefix (voliteľne),
- príkaz,
- parametre príkazu.

Prítomnosť voliteľného prefixu v správe označuje znak „:“ na jej začiatku. Prefix používajú servery na označenie pôvodu správy. Prefix v správe, ktorú dostane klient, podáva informáciu o jej odosielateľovi. Užívatelia pri posielaní správ prefix nepoužívajú.

Príkaz musí byť trojciferné číslo alebo reťazec rozpoznaný ako IRC príkaz. Zoznam všetkých príkazov s ich parametrami popisuje kapitola 4 v [1]. Niektoré často používané budú uvedené v ďalšom texte.

Každý príkaz môže mať 0 až 15 parametrov. Parametre, príkaz a prefix sú od seba navzájom oddelené jednou alebo viacerými medzerami. Posledný parameter môže obsahovať aj medzery. Takýto parameter však musí byť uvedený znakom „:“.

IRC správa má potom formát typu:

```
:prefix príkaz parameter1 parameter2 :posledný parameter
```

Celá IRC správa by nemala mať viac ako 512 znakov, v opačnom prípade IRC server neposkytuje užívateľovi istotu, že spracuje aj znaky prevyšujúce tento limit. Správy pozostávajú z ľubovoľných znakov okrem znakov CR a LF, ktoré slúžia ako oddeľovač správ a znaku NUL, ktorý je z povolených znakov vyňatý, pretože by spôsoboval implementačné problémy. Formálnejší je k dispozicii v podkapitole 2.3.1 v [3].

2.3.2 Prehľad často používaných správ

2.3.2.1 Registrácia spojenia

Prvým krokom pri pripájaní na IRC server je registrácia klienta v IRC sieti. Pre úspešné ukončenie tejto fázy IRC protokol definuje postupnosť 3 príkazov:

- `PASS`,
- `NICK`,
- `USER`.

Správa typu `PASS` nastavuje heslo spojenia. Toto heslo sa musí zhodovať s heslom konkrétneho užívateľa v konfigurácii IRC servera. Táto správa je nepovinná a verejne prístupné servery ju ani nevyžadujú. Má uplatnenie hlavne pri vytváraní spojení medzi dvoma IRC servermi.

Príklad: `PASS secretpass`

Správou NICK si užívateľ volí prezývku (*nickname*), pod ktorou bude v IRC sieti vystupovať. Ak je zvolená prezývka neobsadená a dodržuje pravidlá pre jej výber, IRC server klientovi túto

prezývku pridelí. Platná prezývka začína malým alebo veľkým písmenom anglickej abecedy, za ktorým nasleduje obmedzený počet malých alebo veľkých písmen anglickej abecedy, číslic alebo znak z množiny {'[', ']', '\', '`', '_', '^', '{', '}', '|', '-'}. Klient použije tento príkaz aj v prípade, že si chce zmeniť svoju aktuálnu prezývku na inú.

Správou typu `USER` má 4 parametre a klient touto správou zvolí:

- ident, ktorý sa použije, ak klient nemá spustený ident server,
- počiatočné užívateľské módy,
- tretí parameter je nepoužitý, typicky sa na jeho miesto dosadí znak „*“,
- skutočné meno užívateľa, ktoré môže obsahovať aj medzery.

Vyššie spomenuté užívateľské módy slúžia na obmedzenie príjmu niektorých typov správ, prípadne na signalizáciu určitého stavu klienta ostatným klientom siete. Klient pošle bitový súčet módov, ktoré chce po pripojení nastaviť. Významný je druhý (i) a tretí (w) bit, ostatné sú v tomto prípade ignorované. Nasledujúca tabuľka obsahuje prehľad všetkých užívateľských módov a ich vysvetlenie.

Názov	**Skratka**	**Popis**
away	`a`	Označuje užívateľovu neprítomnosť
invisible	`i`	Užívateľ nie je zahrnutý vo výsledkoch vyhľadávania užívateľov
wallops	`w`	Užívateľ prijíma informačné správy od IRC operátorov
restricted	`r`	IRC server nastavil užívateľovi obmedzené pripojenie; užívateľ nemá možnosť meniť svoju prezývku ani stať sa operátorom kanálu
operator	`o`	Identifikácia globálneho IRC operátora
local operator	`O`	Identifikácia lokálneho IRC operátora
server notices	`s`	Užívateľ prijíma upozornenia z IRC servera

Tabuľka 2.2: Prehľad užívateľských módov

Zmenu užívateľských módov zabezpečuje príkaz MODE, ktorého jediný parameter je zoznam skratiek módov, ktoré chce užívateľ zmeniť. V závislosti na tom, či chce užívateľ daný mód zapnúť alebo vypnúť uvedie pred jeho skratkou znak „+“ alebo „-“. Nie je potrebné aktivačný, resp. deaktivačný znak uvádzať pred každým módom, ak za sebou nasleduje niekoľko módov, ktoré užívateľ súčasne zapína, resp. vypína.

Príklad:
```
MODE nickname +iw
MODE nickname +i-w
MODE nickname -iw
```

2.3.2.2 Udržovanie aktívneho spojenia

IRC server v pravidelných intervaloch kontroluje stav svojich klientov. Na tento účel poskytuje IRC protokol príkaz `PING`. Klient obdrží od servera túto správu, ktorej parametrom je ľubovoľný reťazec, typicky názov IRC servera. V tomto momente je úlohou klienta čo najrýchlejšie odpovedať správou typu `PONG` s tým istým parametrom, ktorý odbržal v predchádzajúcej správe `PING`. Ak tak neučiní, server ho odpojí z IRC siete.

2.3.2.3 Ukončenie spojenia

Klient ukončí IRC reláciu so serverom správou s príkazom QUIT. Táto správa má jeden nepovinný parameter a tým je text, ktorý server pošle ako dôvod ukončenia spojenia na všetky kanály, na ktorých bol klient pripojený spolu s informáciou o jeho odchode z IRC siete. IRC server potvrdí ukončenie spojenia správou typu ERROR a zatvorí sieťové spojenie.

Príklad: `QUIT :working`

2.3.2.4 Posielanie textových správ

Pre prenos užívateľských správ definuje IRC protokol dva možné príkazy: `PRIVMSG` a `NOTICE`. Oba majú rovnakú syntax. Odlišujú sa v tom, že na správy poslané príkazom `NOTICE` sa nesmú posielať automatické odpovede. Prvým parametrom je cieľ správy – prezývka užívateľa alebo názov kanálu. Druhým parametrom je text správy.

Príklad:
```
PRIVMSG nickname :test message
PRIVMSG #channel :test message
NOTICE nickname :notice message
```

2.3.3 Implementačné problémy

2.3.3.1 Použiteľnosť v rozsiahlych sieťach

IRC servery si musia pre zachovanie funkčnosti IRC siete udržovať informácie o všetkých jej serveroch a klientoch. V čase musia tieto informácie IRC servery aktualizovať a preposielať ostatným serverom. S touto potrebou súvisí problém prevádzky rozsiahlej IRC siete. Sieť s veľmi veľkým počtom pripojených serverov má problémy oboznamovať včas servery o všetkých zmenách, ktoré sa v nej dejú (pripojenie, resp. odpojenie klienta, vznik, resp. zánik kanálu, zmena prezývky klienta apod.). S nekompletnými alebo neaktuálnymi informáciami IRC servery nemôžu zaručiť bezproblémové doručenie klientskych správ, preto je žiaduce udržovať počet serverov IRC siete čo najnižší.

2.3.3.2 Konflikty mien

IRC sieť je tvorená spojením niekoľkých serverov. Pri prevádzke siete môže nastať stav, kedy dva servery medzi sebou stratia spojenie. Takýto stav sa v terminológií IRC nazýva „netsplit“. Sieť v takomto prípade naďalej funguje, avšak každá zo strán prerušenej linky nemá možnosť posielať aktualizácie o novom stave siete. V tomto stave sa k obom stranám rozpojenej siete môžu pripájať noví klienti a môže nastať situácia, kedy si dvaja rôzni klienti vyberú rovnakú prezývku. Keďže je sieť v určitom mieste dočasne rozpojená, nemajú servery možnosť overiť duplicitu klientom vybranej prezývky v rámci celej IRC siete, ale len v časti, ktoré je im dostupná. Tento fakt má za následok, že počas stavu *netsplit* sa k obom stranám rozpojenej IRC siete môžu pripojiť klienti s rovnakou prezývkou.

Po znovunadviazaní prerušeného spoja je nutné vyriešiť problém existencie dvoch rovnakých prezývok. V skorších implementáciách IRC servery riešili tento problém buď násilným ukončením spojenia s oboma klientmi (čím sa konfliktná prezývka opäť uvoľnila) alebo zmenou prezývok oboch strán konfliktu na ich unikátne užívateľské identifikátory a zablokovanie ich pôvodnej prezývky na určitý časový interval, po ktorom si môže niektorý z klientov prezývku obsadiť.

V implementáciách IRC serverov sa začali uplatňovať aj techniky, ktoré vzniku duplicitných mien predchádzajú. Dva najznámejšie spôsoby sú:

- Nick/Channel delay (ND/CD),
- Timestamping (TS).

Základný princíp metódy ND/CD je zablokovanie mien (prezývok a kanálov), o ktorých server stratí informáciu pri rozpojení siete (*netsplit*). Ak chce následne niektorý klient takéto meno použiť, server mu oznámi jeho dočasnú nedostupnosť. Po uplynutí časového zámku (typicky 30min.) alebo znovuspojení siete je možné zablokované mená znovu používať.

Pri použití metódy TS je každé meno (prezývka, názov kanálu) zviazané s časovým razítkom jeho vytvorenia. V prípade rozpojenia IRC siete môžu klienti na oboch stranách používať rovnaké prezývky, avšak po znovuspojení siete, je spojenie s klientom, ktorý má novšie časové razítko *(timesptamp)* ukončené spojenie. V praxi sa ukázalo, že táto metóda môže spôsobovať problémy v okamžiku, keď sa po znovuustanovení spojenia IRC servery nevedia dohodnúť na novom stave siete. Preto býva vo väčsine implementácií, pre zvýšenie spoľahlivosti, táto metóda doplnená o mechanizmus podobný ND/CD.

2.3.3.3 Bezpečnosť

Všetky správy sa prenášajú v podobe otvoreného textu. Medzi kritické patria hlavne správy typu PASS a OPER, v ktorých sa prenáša užívateľské, resp. operátorské heslo, ktoré môže byť počas

prenosu odpočúvané a následne zneužité. Tento problém riešia servery vyžitím služieb šifrovaného transportného protokolu TLS/SSL.

3 IRC roboty

IRC robot je program, ktorý z pohľadu IRC siete vystupuje ako normálny IRC klient. IRC robot musí mať implementovanú klientsku časť protokolu IRC. Nepoužíva sa však na interaktívne sprístupnenie systému IRC koncovému užívateľovi, ale na vykonávanie automatizovaných činností a poskytovanie služieb autorizovaným užívateľom.

3.1 Poskytované služby

Účel existencie IRC robotov je rôznorodý. Spravidla implementujú funkcionalitu, ktorú priamo neposkytuje IRC sieť. Nasledujúce podkapitoly poskytujú súpis najčastejších funkcií poskytovaných robotmi.

3.1.1 Štatistiky a záznamy

IRC roboti bývajú často používaní na vytváranie štatistík o klientoch a IRC kanálov. Tieto štatistiky sa typicky týkajú počtu správ, ktoré užívateľ odoslal na kanál. Zo zaznamenaných údajov následne sa počítajú rôzne informácie o konkrétnom užívateľovi:

- priemerný počet slov, resp. písmen v správe,
- priemerná dĺžka správ,
- celkový počet správ,
- aktivita užívateľa v rôznych časového intervaloch počas dňa.

Roboty môžu vytvárať záznamy komunikácie (*logs*) na určitých kanáloch a verejne ich sprístupniť pre užívateľov, ktorí boli odpojení a zaujíma ich história diskusie na kanáli.

Obe vyššie spomenuté funkcie majú pre užívateľov len informatívny charakter a nie sú veľmi dôležité, pretože si podobnú funkcionalitu môžu klienti implementovať samostatne. IRC roboty ju implementujú, pretože sú k IRC sieti väčšinou neustále pripojení a na rozdiel od klientov, ktorí sa odpájajú a pripájajú, majú komplexný prehľad o správach na IRC kanáloch. Ďalším dôvodom pre implementáciu týchto funkcií je, že nie každý IRC klient ich dokáže užívateľovi poskytnúť.

3.1.2 Zábava

Niektoré roboty poskytujú užívateľom IRC kanálu textové hry. Ide hlavne o hry typu kvíz, kde sa užívatelia snažia na základe indícií poskytovaných robotom uhádnuť slovo, resp. slovné spojenie, za ktoré následne získavajú body. Robot si udržuje tabuľku bodového hodnotenia a na konci hry vypíše hodnotenie jednotlivých užívateľov a víťaza. Táto funkcia IRC robotov nevznikla z nedostatku

možností poskytovaných IRC sieťou. Väčšinou sa prevádzkuje na IRC kanáloch zameraných na takúto činnosť, nakoľko pri spustenej hre tohto typu je veľmi neprehľadné udržiavať akúkoľvek inú konverzáciu.

3.1.3 Vyhľadávanie

Menej používanou funkciou niektorých robotov je IRC rozhranie pre webové vyhľadávače (napr.: www.google.com, www.yahoo.com). Použitie je jednoduché, užívateľ si formou IRC správy vyžiada vyhľadanie výrazu na konkrétnom vyhľadávači. Robot sa pomocou protokolu HTTP pripojí k vyhľadávaču, zadá hľadaný reťazec a zobrazí výsledky vyhľadávania späť užívateľovi. Nakoľko sú výsledky zobrazované formou textových správ, robot ich zvyčajne obmedzí len na niekoľko prvých výskytov, aby zbytočne nezaplňoval IRC kanál textom.

3.1.4 Databáza pojmov

Jednou z ďalších funkcií, ktoré IRC roboty ponúkajú je schopnosť zapamätať si vysvetlenie určitého pojmu. Toto užívatelia využívajú na uloženie často používaných URL adries alebo na objasnenie rôznych skratiek.

3.1.5 Správa kanálov

Najdôležitejšiu úlohu IRC roboty zohrávajú pri správe kanálov. Potreba použitia robotov pri správe kanálov vzniká preto, že kanály sú z podstaty IRC protokolu dynamické. To znamená, že vznikajú pripojením prvého užívateľa, ktorý automaticky získava právo operátora kanálu a zanikajú odchodom posledného užívateľa. Ak má teda užívateľ záujem o trvalé „vlastníctvo“ určitého IRC kanálu, musí si zaistiť, aby zostal neustále otvorený. Toto nie je pre užívateľa celkom jednoduché, pretože väčšina užívateľov svoje počítače viac či menej pravidelne vypína. V tomto prípade môže užívateľ pripojiť na daný IRC kanál robota, ktorý je spustený na neustále zapnutom počítači a teda drží kanál otvorený aj po odpojení všetkých ostatných klientov.

Ďalším problémom, ktorý vzniká pri potrebe udržania vlastníctva IRC kanálu je obnova práva operátora kanálu autorizovaným užívateľom. Po odpojení užívateľa z IRC siete server odstráni zo svojich dátových štruktúr všetky informácie o ňom. V týchto informáciách sú zahrnuté aj práva na všetkých kanáloch a jeho užívateľské módy. Po znovu pripojení užívateľa do IRC siete nemá IRC server prostriedky na zistenie, či ide o užívateľa, ktorý sa pred časom odpojil, alebo ide o nového užívateľa s tou istou identifikáciou (prezývka, ident, hostname). Preto nie sú užívateľom opätovne pridelené ich predchádzajúce právomoci na kanáloch. Ak chce takýto užívateľ na svojom kanáli opäť získať právo operátora kanálu, musí mu ho udeliť niekto z existujúcich operátorov. Ak je jedným z nich IRC robot je možné túto činnosť automatizovať tým spôsobom, že robot po overení totožnosti

užívateľovi pridelí práva, na ktoré má nárok. O možnostiach, ktorými IRC roboty autentizujú užívateľov hovorí podkapitola 3.1.5.2.

3.1.5.1 Rozšírené užívateľské práva

IRC protokol poskytuje tri úrovne právomocí na kanále:

- operátor,
- užívateľ s právom *voice*,
- normálny užívateľ.

V niektorých prípadoch môže byť žiaduce jemnejšie delenie právomocí. Napríklad zmenu témy kanálu môžu v závislosti na jeho nastavení meniť buď len operátori alebo všetci užívatelia. Protokol neposkytuje prostriedky na vyčlenenie skupiny užívateľov, ktorý majú len právo meniť tému kanálu. Podobný problém vzniká pri mnohých úkonoch, ktoré môžu na kanále vykonávať iba jeho operátori, ale nedajú sa delegovať na užšiu skupinu užívateľov bez toho, aby dostali neobmedzené právo operátora kanálu.

Keďže dať právo operátora kanálu širšej skupine užívateľov predstavuje bezpečnostné riziko, používajú sa aj v tomto prípade IRC roboty. Tie majú implementované dostatočné delenie právomocí. Každému autorizovanému užívateľovi povolí robot vykonávať len tie úkony, na ktoré má získal od majiteľa robota oprávnenie.

Všetci roboti poskytujú užívateľské práva, ktoré minimálne zodpovedajú tým, ktoré definuje protokol IRC (kanálový operátor, užívateľ s právom *voice*, normálny užívateľ). Nad rámec tohto základu ponúkajú užívateľom ďalšie možnosti, ktoré má z pohľadu IRC protokolu len operátor kanálu. Ide napríklad o už spomínané nastavovanie témy kanálu, pozývanie klientov na kanál, alebo nastavovanie niektorých atribútov kanálu. Vďaka tomu môže majiteľ IRC robota preniesť časť svojich kompetencií na vymedzenú skupinu užívateľov bez toho, aby mali právo operátora kanálu. Týmto majiteľ robota a vlastník kanálu eliminuje riziko zámerného alebo neúmyselného narušenia fungovania kanálu, nakoľko užívatelia s určitým delegovaným právom nie sú priamo operátormi kanálu, ale činnosť, na ktorú sú autorizovaní vykonávajú prostredníctvom IRC robota.

3.1.5.2 Možnosti autentizácie

IRC roboty zvyčajne implementujú 3 možnosti autentizácie užívateľov:

- autentizácia menom a heslom,
- autentizácia identifikáciou na IRC,
- kombinácia predošlých dvoch metód.

Overenie na základe užívateľského mena a hesla spočíva v tom, že užívateľ pošle robotovi IRC správu, ktorá obsahuje údaje, ktorými sa chce autentizovať. Robot následne overí správnosť autentizačných údajov a v prípade zhody povolí užívateľovi formou IRC správ vykonávať úkony,

ktoré zodpovedajú jeho úrovni oprávnení na danom kanále. Užívateľ posiela svoje meno a heslo typicky správou typu `PRIVMSG`, ktorú adresuje priamo IRC robotovi.

Táto metóda má vážny bezpečnostný nedostatok. IRC správy sa väčšinou prenášajú v podobe otvoreného textu, a preto je možné ich obsah (v tomto prípade užívateľské meno a heslo) odpočúvať a následne zneužiť. Čiastočným riešením tohto problému je použitie šifrovaného spojenia k IRC serveru. Príjemca správy však túto možnosť nemusí vyžívať. V rámci IRC siete sa správa preposiela nešifrovaná, preto úplné zabezpečenie proti odposluchu mena a hesla neexistuje.

Ďalšou možnosťou overenia užívateľa je využitie identifikácie, ktorú má v rámci IRC siete: prezývka, ident, hostname. Z tých údajov sa zvyčajne používajú len posledné dva, nakoľko existencia prezývok je dynamická. Tento spôsob overenia je pre užívateľa pohodlnejší pretože nemusí posielať robotovi meno a heslo a nevystavuje sa tak riziku ich odposluchu. Nevýhodou tejto metódy je jej obmedzená použiteľnosť pre užívateľov, ktorý sa pripájajú zo sietí s dynamickým prideľovaním IP adries a pre užívateľov, ktorí nemajú zaručenú pravdivú identifikáciu TCP spojenia s IRC serverom (ident). Užívatelia, ktorí sa pripájajú do IRC siete z niekoľkých počítačov, môžu túto možnosť využiť tiež, nakoľko väčšina robotov umožňuje asociovať s užívateľom viac ako jednu IRC masku.

3.2 Prehľad existujúcich implementácií

S rozšírením IRC protokolu postupne vznikli desiatky IRC robotov, ktoré sa od seba odlišujú zameraním, množstvom ponúkaných funkcií a platformou, na ktorej sú spustiteľní. Z dôvodu prehľadnosti preto v nasledujúcom texte spomeniem len tie implementácie, ktoré sa celosvetovo rozšírili a licencia, pod ktorou sú vydaní povoľuje slobodné používanie vrátane možnosti obdržania zdrojových súborov týchto programov.

V posledných rokoch sa v IRC komunite veľmi rozšíril IRC robot Eggdrop. Je napísaný v jazyku C, poskytuje dosť širokú škálu funkcií a je univerzálne použiteľný. Jeho nevýhodou boli vážne bezpečnostné chyby v skorších verziách a pomerne komplikovaná konfigurácia, najmä pre neskúsených užívateľov. Rozšíriteľný je pomocou užívateľských skriptov napísaných v jazyku TCL.

Ďalším z používaných robotov súčasnosti je Psotnic. Tento projekt si kladie za snahu poskytnúť užívateľom stabilný a bezpečný program. Je vyvíjaný pre použitie v sieti IRCnet. Je napísaný v jazyku C++ a umožňuje a dynamické linkovanie užívateľských modulov.

Energymech je posledným zo súčasných často používaných robotov. Je napísaný v jazyku C a kladie dôraz na nízku spotrebu systémových prostriedkov a portabilitu. Najnovšia verzia umožňuje rozšírenie funkcionality pomocou skriptov napísaných v jazykoch Perl, Python a TCL.

Všetky spomínané implementácie tiež podporujú spájanie viacerých robotov do siete, ktorej sa v ich terminológií hovorí *botnet*.

4 Návrh IRC robota

Pri návrhu implementácie IRC robota je treba zohľadniť riešenie niekoľkých úloh. Riešenie návrhu takéhoto programu je teda možné rozdeliť na niekoľko samostatných oblastí. V každej oblasti navrhnúť riešenie a vo výsledku spojiť riešenia čiastkových podproblémov do jedného fungujúceho celku.

Keďže ide o klientsky program, je potrebné implementovať klientsku časť protokolu IRC. IRC je aplikačný protokol, ktorý využíva služby spoľahlivého transportu dát – protokolu TCP. Z toho dôvodu je treba navrhnúť spôsob komunikácie cez TCP/IP sieť.

IRC roboty užívatelia vyhľadávajú, pretože poskytujú služby, ktoré nie sú štandardne zahrnuté v systéme IRC, preto je nutné vymedziť množinu potrebných služieb a navrhnúť spôsob ich implementácie v rámci programu.

Aby mohli užívatelia prispôsobiť program, konkrétnej situácií, v ktorej ho chcú použiť musia mať nástroj na jeho konfiguráciu. Možnosti konfigurácie by mali byť prehľadné a jednoduché na použitie. V opačnom prípade to môže vyvolať najmä u neskúsených užívateľov negatívny dojem z používania programu.

V nasledujúcich podkapitolách objasním, akým spôsobom som sa rozhodol vyššie uvedenú problematiku riešiť, príp. uvediem iné možné riešenia. Pri riešení som sa snažil vytvoriť objektovo orientovaný návrh. V niektorých miestach sa však vyskytujú znaky procedurálneho návrhu.

4.1 Transport dát cez TCP/IP sieť

Aplikačné rozhranie pre prístup k službám transportnej vrstvy poskytuje operačný systém, preto nie je nutné v tomto smere nič navrhovať ani implementovať. Veľmi často používané a rozšírené je rozhranie schránok (*BSD sockets*). Toto aplikačné rozhranie je dostupné na väčšine operačných systémov odvodených od UNIXu. Má podobu nízkoúrovňových systémových volaní. Z toho dôvodu nezapadá do objektovo orientovaného návrhu programu a je potrebné vytvoriť nad ním objektové rozhranie.

V tejto etape návrhu som sa, vzhľadom na vyššie popísanú situáciu, rozhodol vytvoriť hierarchiu tried, ktoré budú poskytovať prístup k rôznym typom komunikácie s vyžitím služieb transportnej vrstvy. Základom tejto hierarchie je trieda `Socket`, ktorá obsahuje metódy a atribúty súvisiace so všetkými typmi schránok. Od triedy `Socket` je odvodená trieda `TCPSocket`, ktorá implementuje spoľahlivý prenos dát pomocou transportného protokolu TCP. V prípade potreby by bolo možné odvodiť ďalšie triedy, napr. `UDPSocket` – implementácia nespoľahlivého dátového

prenosu. Nakoľko program tento typ komunikácie nevyužíva, tak som jej návrh a implementáciu vynechal. Zo spomenutej hierarchie tried vystupuje trieda `TCPServerSocket`, ktorá poskytuje prostriedky pre vytvorenie aplikácie typu server. Implementačné detaily popisuje podkapitola 5.1. Názornejší pohľad na túto časť návrhu podáva časť diagramu tried zobrazeného na obr. 4.2.1.

4.2 Protokol IRC

Ako je uvedené v kapitole 2, v súčasnosti popisuje komunikáciu pomocou protokolu IRC niekoľko „štandardov". Pôvodnú verziu protokolu popisuje dokument [1]. Novšie aspekty klientskej časti protokolu ďalej popisujú dokumenty [2] a [3]. Ako už bolo spomenuté, nie všetky implementácie IRC serverov prijali tieto zmeny v rovnakej miere. Z tohto dôvodu som sa rozhodol pri návrhu modulu, ktorý sprostredkuje komunikáciu cez IRC, obmedziť sa len na pôvodnú špecifikáciu protokolu z dokumentu [1]. Tým síce program príde o možnosť využívať niektoré služby IRC sietí, ktoré [1] nešpecifikuje. Na druhej strane bude program použiteľný na väčšine IRC sietí.

O vytvorenie a udržanie relácie s IRC serverom sa spolu starajú tri triedy. Trieda `Message` reprezentuje jednu IRC správu v štrukturalizovanej podobe. Objekty tejto triedy umožňujú jednoduché spracovanie a analýzu IRC správ. Trieda obsahuje atribúty:

- `type` – typ správy,
- `from` – odosielateľ správy,
- `to` – adresát správy,
- `target` – cieľ príkazu správy (len niektoré typy IRC správ),
- `text` – text správy.

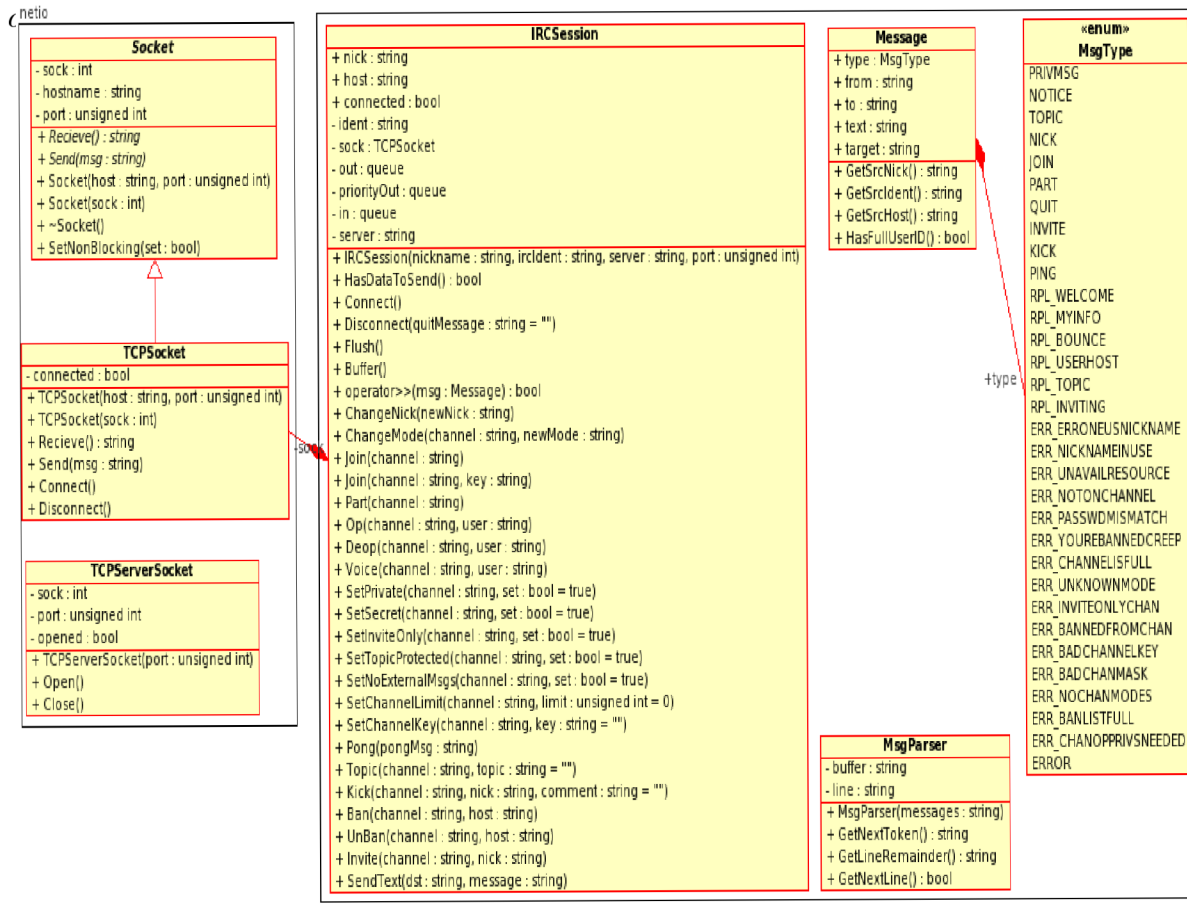
Objekty triedy `Message` vznikajú analýzou správ príjmaných z IRC servera. IRC server ich posiela vo forme otvoreného textu a oddeľuje znakmi ukončenia riadku (CR LF; viď podkapitola 2.3.1). Pri konštrukcii objektov triedy `Message` sa používajú metódy triedy `MsgParser`. Táto trieda postupne z IRC správy reprezentovanej jedným reťazcom znakov oddeľuje jednotlivé zložky, ktoré sa následne použijú pri vytvorení objektu triedy `Message`.

Inicializáciu spojenia, jeho udržovanie, posielanie všetkých typov IRC správ a ukončenie spojenia s IRC serverom implementuje trieda `IRCSession`. Táto trieda obsahuje metódy, ktoré prekladajú požiadavky programu do IRC protokolu a následne odosielajú na IRC server. Metódy triedy `IRCSession` je možné rozdeliť do niekoľkých kategórií:

- IRC relácia: `Connect`, `Disconnect`, `Pong`, `ChangeNick`
- Posielanie a príjem správ: `SendText`, `Flush`, `operator>>`

- správa kanálov: `Join`, `Part`, `Op`, `Deop`, `Voice`, `DeVoice`, `ChangeMode`, `Kick`, `Ban`, `UnBan`, `Topic`, `SetChannelKey`, `SetInviteOnly`, ...

Ucelený zoznam všetkých metód aj s ich prototypmi je možné vidieť v diagrame tried na obr. 4.2.1.



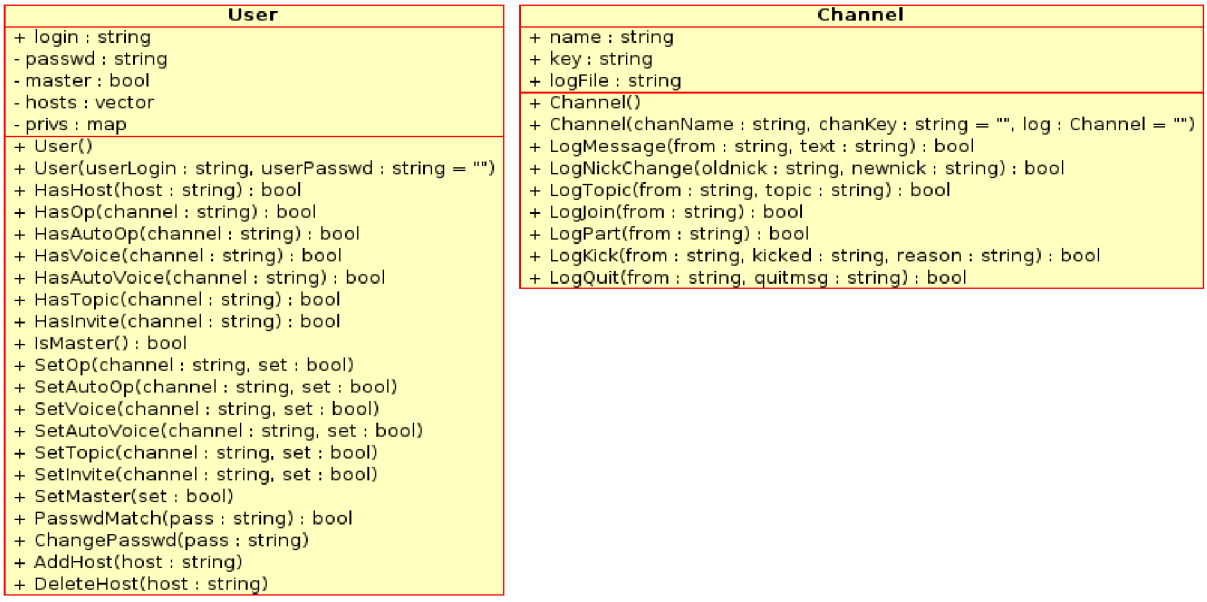
4.3 Funkcionalita

Zo všetkých služieb ponúkaných IRC robotmi (viď podkapitola 3.1) považujem za užívateľsky najpotrebnejšie vykonávanie úloh spojených so správou kanálov, preto som sa rozhodol zamerať na implementáciu tejto funkcionality. Na dosiahnutie tohto cieľa je potrebné navrhnúť dátové štruktúry pre uloženie samotných kanálov, ale aj užívateľov, ich užívateľských rolí a oprávnení. Za týmto účelom som vytvoril dve triedy: `Channel` a `User`.

Trieda `Channel` reprezentuje IRC kanál. Každý kanál je identifikovaný svojím menom (atribút `name`) a môže mať nastavený kľúč pre pripojenie (atribút `key`). Táto trieda tiež obsahuje metódy, ktoré implementujú zápis IRC správ do súboru. S tým súvisí nutnosť ukladať názov súboru, do ktorého sa zapisujú správy adresované danému kanálu (atribút `logFile`). Zápis každého typu správy implementuje jedna metóda, pretože správy rôznych typov sa pred zápisom do súboru odlišne formátujú.

Trieda `User` reprezentuje užívateľa programu. Každý užívateľ je identikovaný prihlasovacím menom a množinou IRC masiek, z ktorých je na IRC sieti rozpoznaný. Ostatné atribúty triedy zohľadňujú užívateľské oprávnenia (atribúty `privs` a `master`) a užívateľské heslo (atribút `passwd`). Užívateľské heslo nie je uložené v podobe otvoreného textu. Z bezpečnostných dôvodov sa ukladá len výsledok jednosmernej transformačnej funkcie MD5 [5]. Metódy tejto triedy poskytujú rozhranie na pridávanie, odoberanie a zisťovanie stavu užívateľských oprávnení a priradených IRC masiek.

Triedy Channel a User sú graficky zobrazené formou diagramu tried so všetkými atribútmi a metódami na obr. 4.3.1.



obr. 4.3.1: Diagram tried 2

4.4 Konfigurácia

Pre prispôsobenie programu konkrétnym požiadavkám užívateľom som sa rozhodol vytvoriť dva nástroje: systém konfiguračných súborov a administračnú konzolu.

4.4.1 Konfiguračné súbory

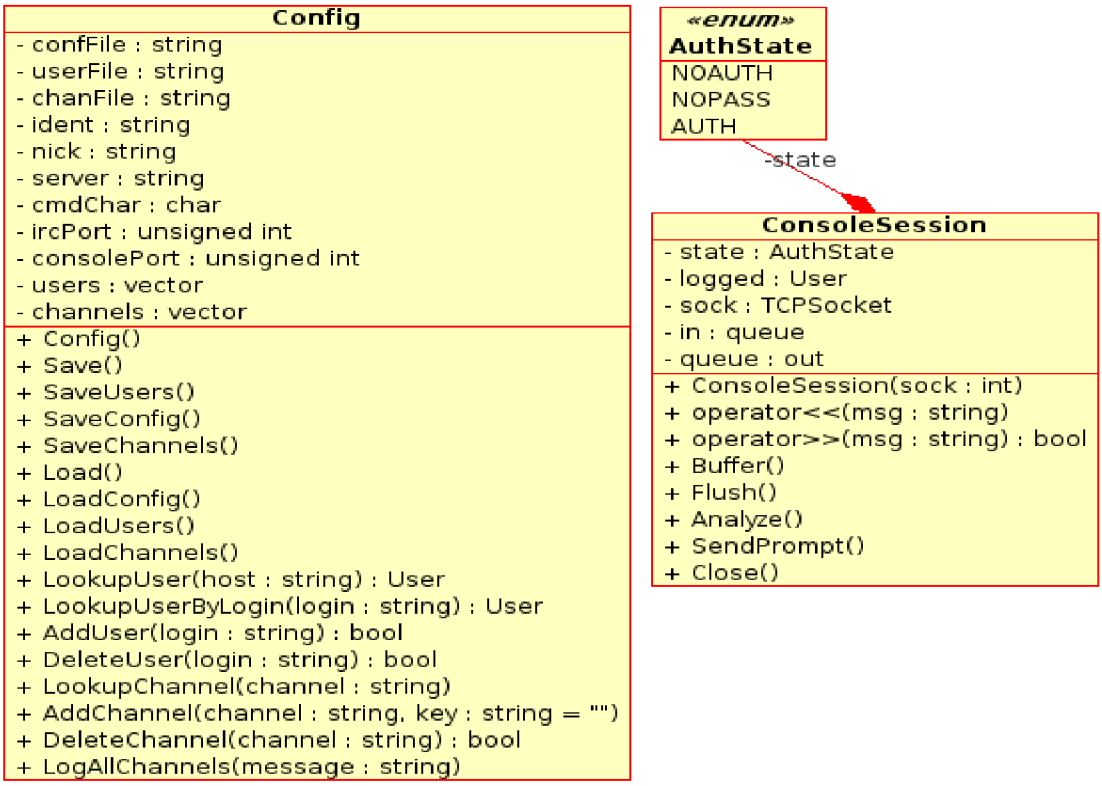
Konfiguračné súbory umožnia užívateľovi ukladať nastavenie programu perzistentným spôsobom. Robot používa na zostavenie svojej konfigurácie tri súbory. Všetky súbory majú podobu otvoreného textu a používajú podobné syntaktické prostriedky na vyjadrenie rôznych aspektov konfigurácie.

Globálny konfiguračný súbor obsahuje nastavenia týkajúce sa najmä pripojenia k IRC sieti a umiestnenie ostatných konfiguračných súborov.

Užívateľský konfiguračný súbor obsahuje definície užívateľov, ktorí majú možnosť využívať služby, ktoré program ponúka. Každý uvedený užívateľ musí mať povinne uvedené prihlasovacie meno a heslo (MD5 hash) a voliteľne oprávnenia a zoznam IRC masiek, z ktorých je rozpoznaný.

Posledný súbor obsahuje konfiguráciu kanálov, na ktoré sa má robot po spustení pripojiť. Každý záznam o kanáli obsahuje povinne jeho meno a voliteľne kľúč nutný pre pripojenie a súbor, do ktorého má program zapisovať správy adresované kanálu.

Načítanie a ukladanie konfigurácie z a do konfiguračných súborov zabezpečuje trieda `Config`. Táto trieda tiež obsahuje metódy určené na zmenu aktuálnej konfigurácie. Atribúty triedy uchovávajú údaje načítané z konfiguračných súborov. Ich zoznam je k dispozícii v diagrame tried na obr. 4.4.1.1.



obr 4.4.1.1: Diagram tried 3

4.4.2 Administračná konzola

Program načítava nastavenia z konfiguračných súborov po spustení. Aby užívateľ nemusel pri každej zmene v konfigurácií reštartovať program, má k dispozícii tzv. administračnú konzolu. Ide o rozhranie, ktoré povoľuje vykonávať zmeny za behu programu bez nutnosti reštartu. Reláciu s užívateľom cez toto rozhranie implementuje trieda `ConsoleSession`. Táto trieda má na starosti všetky fázy komunikácie medzi užívateľom a programom: ustanovenie spojenia, autentizácia, vykonávanie užívateľských príkazov a ukončenie spojenia.

Autentizačný stav klienta si objekt triedy `ConsoleSession` ukladá do atribútu `state`. Tento atribút môže nadobúdať tri hodnoty: neautentizovaný klient (`NOAUTH`), stav so zadaným prihlasovacím menom (`NOPASS`) a autentizovaný klient (`AUTH`).

Komunikácia s klientom je riešená pomocou vstupných a výstupných vyrovnávacích zásobníkov (atribúty `in` a `out`). O naplnenie a vyprázdňovanie týchto dátových štruktúr sa starajú metódy `Buffer`, resp. `Flush`. Analýzu prijatých správ vykonáva metóda `Analyze`. Pre overovanie oprávnení na vykonávanie príkazov, ktoré klient poslal slúži atribút `logged`, ktorý obsahuje odkaz na objekt identifikujúci prihláseného užívateľa (objekt triedy `User`). Diagram triedy `ConsoleSession` sa nachádza na obr. 4.4.1.1.

4.4.3 Aktuálna konfigurácia

Všetky hodnoty aktuálnych nastavení si program ukladá do objektu triedy `Config`, ktorá má atribúty uchovávajúce hodnoty nastavenia: prezývky robota (`nick`), IRC identu (`ident`), adresy IRC servera (`server`), portu IRC servera (`ircPort`), portu pre administračnú konzolu (`consolePort`), zoznam všetkých užívateľov (`users`) a kanálov (`channels`). Pre naplnenie posledných dvoch spomenutých atribútov je nutné, aby program poznal umiestnenie užívateľského (`userFile`) a kanálového (`chanFile`) konfiguračného súboru. Hodnoty ostatných atribútov program získa analýzou globálneho konfiguračného súboru (`confFile`). Postupné načítanie a analýzu všetkých troch konfiguračných súborov zabezpečujú metódy `LoadConfig`, `LoadUsers` a `LoadChannels`. Pre uloženie aktuálnej konfigurácie programu do konfiguračných súborov existujú metódy `SaveConfig`, `SaveUsers` a `SaveChannels`. Popis celej triedy formou diagramu je uvedený na obr. 4.4.1.1.

5 Implementácia

Pre implementáciu programu, ktorého návrh popisuje kapitola 4, som zvolil jazyk C++. Toto implementačné prostredie som si vybral, pretože poskytuje prostriedky pre uplatnenie objektovo orientovaného návrhu s možnosťou využiť prvky procedurálneho a generického programovania. Jazyk C++ tiež umožňuje vývoj modulárnych programov a obsluhu mimoriadnych udalostí formou výnimiek. Pri implementácií som dbal na maximálnu portabilitu programu, preto sú všetky zdrojové kódy programu validné podľa normy ISO/IEC 14882.

Pri implementácií návrhu programu som uplatnil princípy modulárneho vývoja, tzn. každú časť návrhu IRC robota som implementoval do jedného modulu. Tento prístup má niekoľko výhod:

- zdrojové kódy programu sú rozčlenené do logických celkov (prehľadnosť, zrozumiteľnosť),
- pri úprave zdrojového kódu jedného modulu nie je nutné znovu prekladať celý program (úspora času a systémových prostriedkov),
- v prípade potreby jednoduchšiu modifikovateľnosť programu.

V nasledujúcich podkapitolách rozoberiem problémy, ktoré sa pri implementácií vyskytli a uplatnené riešenie, príp. alternatívne riešenia. Vzhľadom na rozsiahlosť implementácie spomeniem len tie časti programu, ktoré sú niečím význačné alebo zaujímavé. Detailný popis implementácie je dostupný formou komentárov v zdrojových textoch programu, ktoré sú priložené na CD.

5.1 Rozhranie schránok (BSD sockets)

Aplikačný protokol IRC využíva pre spoľahlivý prenos dát cez sieť služieb protokolu TCP. Z toho dôvodu je potrebné pri implementácii použiť rozhranie, ktoré služby spoľahlivého prenosu dát sprostredkuje. Rozšíreným a veľmi známym je rozhranie schránok (*BSD sockets*). Toto aplikačné programové rozhranie bolo prvý krát predstavené v operačnom systéme 4.2BSD pre jazyk C. V súčasnosti existuje toto rozhranie vo väčšine operačných systémov.

Schránka *(socket)* vytvára koncový komunikačný bod pre medziprocesovú komunikáciu prostredníctvom počítačovej siete. Z pohľadu operačného systému ide o abstraktnú dátovú štruktúru, ktorá obsahuje údaje potrebné pre komunikáciu cez sieť.

Značnou výhodou rozhrania schránok je možnosť pracovať so schránkou ako s lokálnym súborom. To znamená možnosť používať systémové volania určené na vstup a výstup do a zo súboru *(read ,write)*. Vstupno-výstupné operácie je však nutné realizovať na úrovni tzv. *file descriptorov*. Takéto nízkoúrovňové čítanie a zápis je však nepraktické a nedovoľuje používať vstupno-výstupný systém jazyka C++, preto som nad ním vytvoril „objektovú obálku“. Za týmto účelom som

implementoval hierarchiu tried s bázovou triedou `Socket`, ktorej návrh je v podkapitole 4.2. Trieda implementuje nízkoúrovňové vstupno-výstupné operácie nad schránkami prostredníctvom objektov triedy `std::string`. Po vytvorení týchto operácií môže zvyšok programu čítať a zapisovať zo sieťových schránok použitím objektov štandardnej triedy `std::string`, ktorej hlavná výhoda spočíva v automatickej správe pamäti.

Bázová trieda `Socket` definuje operácie a atribúty spoločné pre všetky typy schránok. Triedy od nej odvodené následne definujú operácie, ktoré súvisia s daným typom schránky, prípadne predefinujú operácie z bázovej triedy. Ide napríklad o metódy `Connect` a `Disconnect` v triede `TCPSocket`. Hierarchia tried je implementovaná v súbore `netio.cpp` s rozhraním `netio.h`.

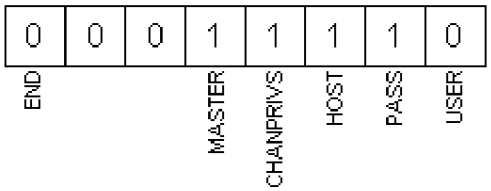
5.2 Analýza konfiguračných súborov

Ako už bolo spomenuté, všetky nastavenia programu uchováva objekt triedy `Config`. Konfigurácia sa po štarte programu inicializuje na implicitné hodnoty a následne sa aktualizuje hodnotami zadanými v konfiguračných súboroch. Umiestnenie globálneho konfiguračného súboru získa program ako prvý a jediný argument. V prípade vynechania tohto argumentu sa pokúsi otvoriť súbor `ircbot.conf` z aktuálneho adresára (*working directory*).

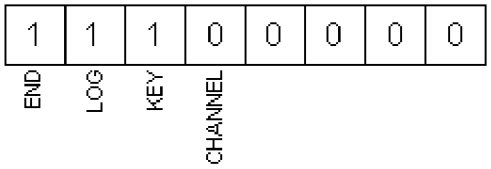
Všetky konfiguračné súbory sú textové. Konfigurácia je zapísaná v dvojiciach slov. Prvé slovo označuje konkrétnu časť nastavenia programu (v priloženej užívateľskej príručke je označované ako direktíva). Druhé slovo vyjadruje hodnotu nastavenia. Analýza súboru prebieha po riadkoch. Každý riadok, ktorý začína znakom „#“ je považovaný za užívateľský komentár a je programom ignorovaný.

Pre analýzu konfiguračných súborov by bolo možné zostaviť konečný automat s množstvom stavov. Ide teda o regulárny jazyk. Konečné automaty sa v procedurálnych jazykoch typicky implementujú pomocou vetvenia `switch`, pričom jednotlivé vetvy predstavujú hrany automatu. Tento spôsob som však pre analýzu jednoduchej syntaxe použitej v konfiguračných súboroch považoval za zbytočne komplikovaný. Pri implementácií sa stav analýzy ukladá do bitového pola. Keďže pri analýze je potrebných len niekoľko stavov, bitové pole je realizované pomocou jednej premennej typu `char`. Tento dátový typ má podľa normy jazyka vždy jeden bajt, teda poskytuje možnosť uloženia hodnôt bitového pola o veľkosti 8. Položky bitového pola predstavujú názvy tých kľúčových slov konfiguračných súborov, ktoré vzhľadom na aktuálny kontext môžu nasledovať. Prístup ku konkrétnemu bitu v programe riešim operáciou bitového súčinu stavu analýzy a bitovej masky, ktorej hodnota je pre konkrétne kľúčové slovo definovaná symbolickou konštantou. Symbolický pohľad na implementáciu stavu analýzu poskytujú obr. 5.2.1 a obr. 5.2.2. Pri načítaní novej slova z konfiguračného súboru sa s využitím vyššie popísaného mechanizmu skontroluje, či je

prečítaná direktíva vzhľadom na kontext analýzy očakávaná. V prípade prečítania neočakávanej alebo nerozpoznanej konfiguračnej direktívy je celý konfiguračný súbor vyhodnotený jako nesprávny. Program na túto situáciu reaguje vyhodením výnimky.



Obr. 5.2.1: Stav analýzy užívateľského konfiguračného súboru



Obr. 5.2.2: Stav analýzy kanálového konfiguračného súboru

Obdobným spôsobom ako pri analýze konfiguračných súborov postupuje program aj pri analýze príkazov, ktoré môžu zadávať užívatelia cez administračnú konzolu.

5.3 Výnimky

Obsluhu mimoriadnych udalostí, ktoré vznikajú za behu programu som implementoval pomocou výnimiek (*exceptions*). Ide o nástroj jazyka C++, ktorý umožňuje jednoducho a prehľadne spracovať chyby na jednom mieste v programe. Nevýhodou použitia tohto mechanizmu je o niečo pomalší beh programu. Túto nevýhodu považujem v súčasnosti za zanedbateľnú v porovnaní s výhodami, ktoré takéto riešenie poskytuje.

Pre spracovanie chýb pomocou výnimiek som navrhol a implementoval hierarchiu niekoľkých tried, ktoré sú odvodené od bázovej triedy `Exception`. Táto trieda má atribút `msg`, ktorý uchováva chybovú správu. Výpis chyby na štandardný chybový výstup zabezpečuje metóda `PrintError`.

Každá z odvodených tried predstavuje jeden typ chyby, ktorá sa môže vyskytnúť počas behu programu:

- `Exception` – bázová trieda,
- `Restart` – reštartovať program, znovu načítať konfiguráciu,
- `NetSocketClosed` – schránka (*socket*) bola zatvorená klientom,
- `NetSocketReadError` – chyba pri čítaní dát zo schránky,
- `NetSocketWriteError` – chyba pri zápise dát do schránky,

- `NetSocketError` – nebolo možné vytvoriť sieťovú schránku,
- `NetLookupError` – nebolo možné vyhľadať doménové meno,
- `NetConnectError` – nebolo možné pripojiť sa k vzdialenému procesu,
- `NetBindError` – chyba pri prepojení (*bind*) na lokálny port,
- `IOFileOpenError` – chyba pri otváraní konfiguračného súboru,
- `ConfigUnknownDirective` – nesprávna syntax konfiguračného súboru,
- `ConsoleBadLogin` – pri prihlasovaní do administračnej konzoly bolo zadané zlé meno alebo heslo.

Väčšina výnimiek sa obsluhuje v module, ktorý implementuje funkciu main, tj. `main.cpp`. Výnimky, z ktorých sa program nedokáže zotaviť (`NetSocketReadError`, `NetSocketError`, `NetBindError`) spôsobia ukončenie programu a výpis dôvodu ukončenia na štandardný chybový výstup.

5.4 Automatický preklad

Pre preklad všetkých modulov programu a ich následné linkovanie som sa rozhodol použiť systém make. Tento program na základe pravidiel uvedených v súbore Makefile postupne preloží celý projekt. Keďže sa operačné systémy, na ktorých som program vyvíjal (Linux, FreeBSD, Solaris) líšia v spôsobe linkovania knižníc súvisiacich s rozhraním BSD sockets, vytvoril som jeden generický makefile, ktorý obsahuje pravidlá pre preklad modulov a pre každý operačný systém ďalší makefile, ktorý definuje knižnice, ktoré sa majú k programu prilinkovať.

Pri preklade projektu stačí spustiť v adresári so zdrojovými súbormi program GNU Make, ktorý v závislosti na operačnom systéme použije jeden z pripravených súborov Makefile.

V súboroch makefile sú definované pravidlá pre preklad projektu pomocou prekladaču g++ z balíka GCC (GNU Compiler Collection). Zdrojové kódy programu zodpovedajú norme jazyka C++ z roku 1998 a preto by nemal byť problém použiť v prípade potreby aj iný prekladač. V takom prípade je však nutné prispôsobenie súborov Makefile.

6 Záver

V tejto práci som sa venoval protokolu IRC, jeho použitiu a zjednodušeniu úkonov súvisiacich s používaním systému IRC pomocou IRC robotov. Následne som navrhol a implementoval IRC robota vrátane jeho administračného rozhrania.

Program je v súčasnom stave pre svoj účel použiteľný. Z užívateľského hľadiska považujem za dobre spracované možnosti konfigurácie pomocou prehľadných textových súborov ako aj pomocou administračnej konzoly.

IRC robot bol otestovaný na operačných systémoch FreeBSD 6.3, SunOS 5.7 (Solaris) a Linux 2.6. Prenositeľnosť na iné posixové operačné systémy by nemala byť problematická, nakoľko program používa len štandardné knižnice jazykov C/C++ a štandardné posixové systémové volania. Platforma Microsoft Windows poskytuje rozhranie pre komunikáciu cez sieť nekompatibilné s inými posixovými operačnými systémami, a preto nie je v súčasnej podobe programu prenositeľnosť na túto platformu implementovaná. Avšak vďaka vrstvovému návrhu architektúry programu je možné tento cieľ v budúcnosti dosiahnuť úpravou modulov implementujúcich sieťovú komunikáciu.

Okrem spomenutej prenositeľnosti na platformu Windows by bolo v budúcnosti možné do programu pridať podporu pre spracovanie prichádzajúcich správ užívateľskými skriptami. Vhodné by bolo použiť interpret niektorého v súčasnosti rozšíreného skriptovacieho jazyka – Perl, príp. Python. Túto možnosť by pravdepodobne ocenili hlavne užívatelia, ktorí nemajú znalosť architektúry programu, príp. programovacích jazykov C/C++, a chýba im niektorá špecifická funkcia.

Pre užívateľov, ktorí majú potrebu mať spustených niekoľko inštancií programu by mohlo byť v budúcnosti vhodné implementovať vzájomnú komunikáciu robotov a vzájomné zdieľanie niektorých častí konfigurácie. Inou alternatívou by bolo ukladanie a načítavanie konfigurácie z databázového servera, čo by tiež zjednodušilo správu viacerých IRC robotov.

Literatúra

[1] OIKARINEN, J., REED, D. *Internet Relay Chat* [Online] May 1993. [cit. 2008-04-30] Dostupné z WWW: <http://tools.ietf.org/html/rfc1459>

[2] Kalt, C. *Internet Relay Chat: Architecture* [Online] April 2000. [cit. 2008-04-30] Dostupné z WWW: <http://tools.ietf.org/html/rfc2810>

[3] Kalt, C. *Internet Relay Chat: Client Protocol* [Online] April 2000. [cit. 2008-04-30] Dostupné z WWW: <http://tools.ietf.org/html/rfc2812>

[4] Johns, M. St. *Identification protocol* [Online] February 1993. [cit. 2008-04-30] Dostupné z WWW: <http://tools.ietf.org/html/rfc1413>

[5] Rivest, R. *The MD5 Message-Digest Algorithm* [Online] April 1992. [cit. 2008-04-30] Dostupné z WWW: <http://tools.ietf.org/html/rfc1321>

Zoznam príloh

Príloha 1. CD s elektronickou verziou technickej správy, zdrojovými textami programu a užívateľskou príručkou k programu