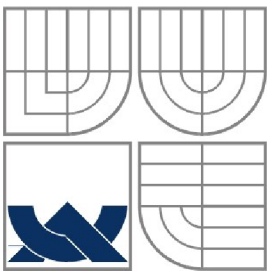
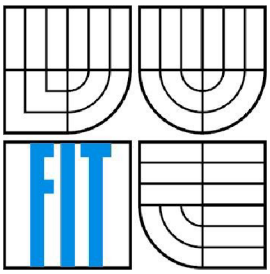


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO SPRÁVU HERNÍCH SERVERŮ

GAME SERVER ADMINISTRATION SYSTEM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — JAN FAIKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. TOMÁŠ KAŠPÁREK
SUPERVISOR

BRNO 2010

Abstrakt

Tato bakalářská práce pospisuje způsoby konfigurace a správy nejpoužívanějších herních serverů, dále pak návrh a implementaci systému ve formě klient - server aplikace určený k jejich spouštění, konfiguraci a ovládání. Součástí je také popis několika nástrojů používaných pro konfiguraci a správu herních serverů. V práci jsou popsány všechny podstatné rozdíly mezi herními servery různých her důležité pro návrh tohoto univerzálního systému. Do srovnání byly zahrnuty herní servery postavené na GoldSource, Source, Id tech, Unreal a TrackMania engine. Součástí je také popis a demonstrace rožšíření systému o novou hru.

Abstract

This bachelor thesis describes configuration and management of the most used game servers, as well as analysis and implementation of the client-server application architecture to control the server. Also some of the tools used to control and configure game server is included. Major differences among various games and servers important for the design of this general tool are described in details. Some of the servers built upon the GoldSource, Source, Id tech, Unreal or TrackMania engine are included into comparison. Important part of the comparison is the description and an example of creating a new extension for a game.

Klíčová slova

Herní server, GoldSource engine, Source engine, Id tech engine, Unreal engine, TrackMania engine, Ruby, protokolový návrh, TCP klient – server, přenos souboru, administrace, vzdálená správa

Keywords

Game server, GoldSource engine, Source engine, Id tech engine, Unreal engine, TrackMania engine, Ruby, protocol design, TCP client – server, file transfer, administration, remote control

Citace

Jan Faika: SYSTÉM PRO SPRÁVU HERNÍCH SERVERŮ, bakalářská práce, Brno, FIT VUT v Brně, 2010

SYSTÉM PRO SPRÁVU HERNÍCH SERVERŮ

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Kašpárka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jan Faika
18. 5. 2010

Poděkování

Touto cestou bych chtěl poděkovat svému vedoucímu panu Ing. Tomáši Kašpárkovi za hodnotné rady a odborné vedení během mé práce.

Obsah

Seznam obrázků

1 Úvod

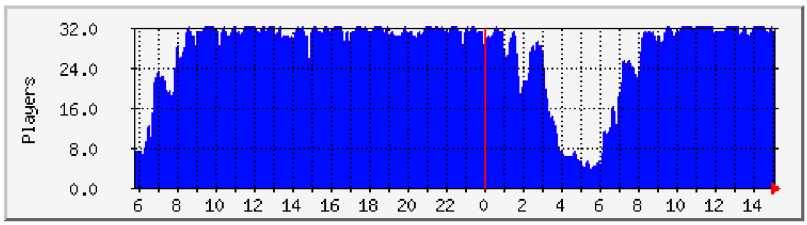
Hraní počítačových her se s postupem času stává stále populárnější. Díky obrovské expanzi vysokorychlostního internetu je také stále větší zájem o hraní her po síti s ostatními lidmi. Valná většina her k propojení více lidí na síti používá klient - server architekturu. Herní servery bývají často velmi náročné na hardwarové prostředky a síťový provoz, výrobci her si tedy nemohou dovolit provozovat dostatečný počet vlastních herních serverů pro uspokojení poptávky. Z toho důvodu jsou u většiny her tyto serverové aplikace volně dostupné a je možné je bezplatně stáhnout z webových stránek. Problémem je, že jen malá hrstka z hráčů, jenž by si ráda spustila server vlastní, ví či se chce naučit jak nainstalovat, nastavit a spravovat herní server. Nastavení herního serveru je zpravidla dosti složité, serverové aplikace jednotlivých her se od sebe dosti liší a pro kompletní správu je potřeba hned několik nástrojů a znalosti k jejich používání.

Hlavním cílem této práce je nastudovat způsoby konfigurace a správy nejpoužívanějších herních serverů, navrhnout a následně implementovat systém ve formě klient - server aplikace určený k jejich spouštění, konfiguraci a ovládání. Důležitý bude návrh komunikačního protokolu, který musí být dostatečně flexibilní. V práci jsou popsány všechny podstatné rozdíly mezi herními servery různých her důležité pro návrh tohoto univerzálního systému. Součástí je také popis a demonstrace rožšíření o novou hru. Systém by měl nalézt využití u poskytovatelů herních serverů, je navržen pro použití s webovým klientem, jenž byl vytvořen v rámci bakalářské práce pana Tomáše Cigána [1].

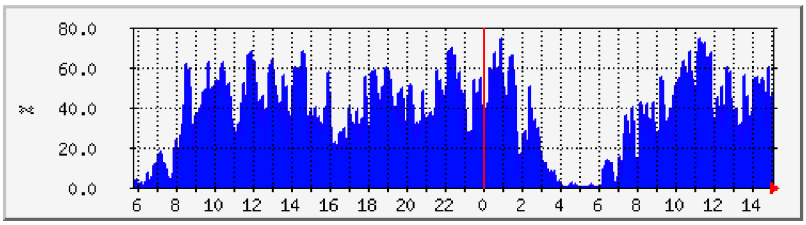
2 Úvod do problematiky

V úvodu bylo zmíněno, že herní servery jsou velice náročné na hardware. Toto tvrzení dokazují grafy na obrázcích 2.1, 2.2 a 2.3. V grafech je zaznamenán průběh počtu hráčů, využití operační paměti a procesorového jádra v závislosti na čase. Jedná se o mírně upravené základní nastavení herního serveru pro hru Counter-Strike 1.6. Tento server běží na hardwarové sestavě Supermicro SuperServer 6015C-MTB, 2 x Intel Xeon E5420 (QuadCore 2,5GHz, 12MB cache 1333MHz FSB), 6 x 2GB RAM 667MHz DDR2 ECC Kingston Value, 2 x Samsung 250GB 7200ot/min - RAID 1. Grafy byly pořízeny pomocí utility MRTG.

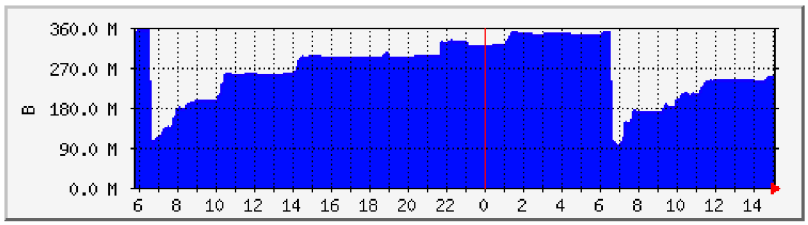
Vzhledem k tomuto faktu není vhodné umisťovat herní server na svůj osobní počítač domů. V případě, že nemáte doma optickou internetovou přípojku, by se navíc dalo očekávat, že by takovýto server byl nekvalitní po stránce ztrátovosti dat a odezvy ke klientům. Z kvalitativního hlediska by bylo nejlepší volbou umístění vlastního hardwarového serveru do datacentra s připojením do páteřní optické sítě, to ale s sebou přináší vysoké náklady a potřebu naučit se herní server kompletně spravovat. Kompromisem mezi těmito možnostmi je objednání serveru u některého z poskytovatelů těchto služeb. Touto možností se budeme zabývat v naší práci.



Obrázek 2.1: Počet hráčů



Obrázek 2.2: Vytížení procesorového jádra



Obrázek 2.3: Využití operační paměti

2.1 Služby a nástroje potřebné k ovládání a správě herního serveru

Tato podkapitola uvádí některé z nástrojů/služeb potřebných k nastavení a správě herních serverů umístěných na vzdáleném počítači (např. server umístěný v datacentru). Naším cílem bude zahrnout do naší aplikace nejdůležitější funkce služeb a nástrojů uvedených v následujících podkapitolách.

2.1.1 SSH (Secure SHell)

Název programu, ale také zabezpečeného komunikačního protokolu s jejichž pomocí je možné navázat šifrované spojení mezi dvěma počítači. Cílem SSH bylo nahradit nijak nezabezpečený telnet. SSH umožnuje vzdálený přístup k příkazové řádce (terminálu) či obecně jakékoliv datové přenosy (např. přenos souborů). Na vzdálený server se můžeme skrze SSH protokol připojit například s programem PuTTY, jenž je dostupný pro operační systémy Windows a Linux. Snad žádný poskytovatel herních serverů SSH přístup z bezpečnostních a jiných důvodů neumožňuje.

2.1.2 Screen

Utilita Screen je celoobrazovkový správce oken, který přepíná fyzický terminál mezi virtuálními. V takovémto virtuálním terminálu se většinou spouštějí procesy umožňující interakci s uživatelem, čímž herní server většinou bývá. Screen v podstatě umožňuje spuštění a ovládání teoreticky neomezeného množství herních serverů. Screen umožňuje logování výpisu herního serveru do souboru, což usnadňuje ladění konfigurace. Screen se spouští z terminálu. Abychom jej mohli použít, musíme mít k terminálu přístup (přímý nebo vzdálený, například přes SSH). Informace byly převzaty ze zdroje [2].

2.1.3 FTP

Vzhledem ke skutečnosti, že SSH přístup nenabízí téměř žádný poskytovatel herních serverů, je zpravidla možné nahrávat, stahovat a mazat soubory herního serveru skrze FTP (síťový protokol určený převážně k přenosu souborů, z angl. File Transfer Protocol). Pro tento způsob správy je potřeba FTP klient nebo jiný software s integrovaným FTP klientem umožňující práci se soubory.

2.1.4 Textový editor

Veškeré nastavení herního serveru bývá uloženo v konfiguračních souborech různého formátu (čistý text, XML). Editace souborů nezkušenými uživateli bývá riskantní, špatná syntaxe v XML souboru mívá za následek nefunkčnost herního serveru. Je vhodné používat pokročilý editor s podporou více kódování a integrovaným FTP klientem.

2.1.5 Query software

Jedná se o program, který se dotazuje herního serveru skrze *query*[1] protokol na jeho stav a nastavení. Příkladem takového software je HLSW Gameserver Tool. Nevýhodou je skutečnost, že v případě neočekávaného ukončení či neúspěšného spuštění serveru jsou ztraceny veškeré informace. Tohoto

1 query – z angličtiny dotaz, skrze query protokol je možné získávat informace o aktuálním stavu herního serveru

protokolu je využito v klientské verzi spolupracující s naším systémem vyvíjené panem Tomášem Cigánem [1].

2.1.6 RCON protokol

Jde o vzdálené ovládání (z angl. Remote CONtrol) neboli protokol určený k vzdálenému ovládání herního serveru. Implementace tohoto protokolu se u různých her liší. Komunikace přes tento protokol nebývá nijak šifrována, využívání vyžaduje autorizaci. Autorizace se provádí ověření hesla. Také touto možností správy se zabývá webová verze klienta našeho systému vyvíjená panem Tomášem Cigánem [1]. Ovládání skrze RCON protokol umožňuje aplikace HLSW zmíněná v podkapitole 2.1.5. Obrovskou nevýhodou je skutečnost, že se aplikované změny nezapisují do konfiguračních souborů herního serveru. Změny proto po restartu serveru pozbydou platnosti.

2.1.7 Herní klient

Do většiny herních klientů bývá zabudována podpora RCON protokolu (v případě, že tuto možnost správy herní server umožňuje) viz podkapitola 2.1.7. Herního klienta také potřebujeme pro ověření správné funkčnosti herního serveru.

2.2 Výběr her

Informace v celé podkapitole byly převzaty z [3]. Bohužel není možné do této práce zahrnout veškeré herní servery, budeme se tedy zabývat jen těmi nejpoužívanějšími. Počítačové hry můžeme rozčlenit na několik žánrů. Abychom navrhli opravdu univerzální systém, stručně uvedeme herní žánry a k nim informace o dostupnosti online hraní.

Adventura (z angl. adventure)

Jedná se o žánr, který je rozvláčný a bývá příběhově zajímavý a spletitý. Cílem hry je splnit hlavní úkol, rozdělený většinou na velké množství podúkolů. Hra je určena pro jednoho hráče, tímto typem her se tedy nebudeme dále zabývat.

Akční počítačová hra

Hlavní náplní akčních počítačových her je eliminace cílů pomocí bojových technik, většinou se jedná o hru v první osobě neboli FPS (z angličtiny First Person Shooter). Až na pár výjimek všechny nově vytvořené hry umožňují hru více hráčů online. Převážně tohoto žánru se bude týkat obsah práce.

Arkáda

Je to žánr počítačové hry, založený na jednoduchém konceptu. Takovéto tituly většinou obsahují kola (úrovně) se zvyšující se obtížností. Hlavním rysem arkádových her je změkčení fyzikálních zákonů. Toto je prováděno s účelem zlepšení hratelnosti.

Možnost společné hry více hráčů se zde objevuje, my se budeme zabývat pravděpodobně nejhranějším titulem z tohoto žánru, který je možné hrát online a přenos informací mezi hráči je založen na klient - server architektuře. Jde o závodní hru Trackmania Nations Forever.

Strategie

Jde o hry, ve kterých ovládáme větší množství objektů a musíme s nimi strategicky nakládat tak, abychom utrpěli co nejmenší újmu na vitalitě a dalších vlastnostech objektů a zároveň způsobili újmu protivníkovi.

Herní servery tohoto žánru nejsou zas až tak rozšířené či jsou implementovány přímo v herním klientu a neexistují dedikované verze, nebudeme se jimi tedy zabývat.

Simulátor

Název žánru sám o sobě napovídá, čím se zabývá. Hry se snaží simulovat situace a úkony spojené s reálným životem. Herními servery pro simulátory se také nebudeme zabývat, v případě, že existují, nemívají téměř žádné nastavení.

Hra na hrdiny (RPG, z angl. Role-playing game)

RPG je druh hry, kde hráči zaujímají role fiktivních postav (role-playing – hraní v roli), které si podle daných pravidel vytvoří a za které v samotné hře jednají. Online hraním se zabývá podskupina MMORPG (Massive multiplayer online role play game - masivně multiplayerová online hra na hrdinu). Herními servery pro MMORPG se nebudeme vůbec zaobírat, takové herní servery mívají obrovské hardwarové nároky a bývají rozloženy na několik hardwarových serverů (databázový server, server komunikující s klienty, aj.). Tyto servery si navíc společnosti v mnoha případech spravují samy a herní server nemusí být vůbec volně dostupný (např. nejpopulárnějších hra z této kategorie World of Warcraft).

3 Druhy herních serverů

Tato kapitola pojednává o v dnešní době nejpopulárnějších herních engine[2]. V kapitole jsou detailně rozebrány všechny jejich podstatné rozdíly důležité pro návrh naší aplikace. V této kapitole není zmíněn také hojně používaný Battlefield engine, to především z toho důvodu, že pro hry postavené na tomto engine jsou vyvíjeny speciální administrační aplikace, které společnost Electronic Arts dle pravidel užívání jako jediné schvaluje pro úpravy těchto serverů. Tyto informace byly čerpány z publikací [4] a [5].

Způsob značení

`HOME`	značí adresář, ve kterém je herní server nainstalován
`<hodnota>`	zastupuje skutečně zadanou hodnotu spouštěcího parametru
`[parametr]`	reprezentuje hodnotu spouštěcího parametru „parametr“ (např. `[exec]` reprezentuje hodnotu parametru `+exec`, `[fs_game]` reprezentuje hodnotu parametru `+set fs_game`)

3.1 GoldSource engine

Ačkoliv je dosti zastaralý, jedná se pravděpodobně o aktuálně nejpoužívanější engine, to hlavně díky popularitě akční hry Counter-Strike, která je známá svou výbornou hratelností. Pro herní server této hry navíc existují tisíce přídavných pluginů, což jí na popularitě ještě přidává. S pomocí GoldSource engine jsou vytvořeny například tyto hry:

- Half-Life (1998)
- Team Fortress Classic (1999)
- Half-Life Opposing Force (1999)
- Counter-Strike (2000)
- Gunman Chronicles (2000)
- Ricochet (2000)
- Deathmatch Classic (2001)
- Day of Defeat (2003)

3.1.1 Spouštění a konfigurace herního serveru

V podkapitole bylo čerpáno ze zdroje [6]. U dedikovaného serveru je již přiložen *shell* skript hlds_run. Tento skript ošetřuje nastavení implicitních hodnot paramterů v případě jejich nenastavení skrze spouštěcí parametry.

Nejdůležitější spouštěcí parametry

`-game <řetězec> (základní hodnota: valve)`
Parametr `-game` určuje hru, pro kterou se má server spustit. Při požadavku spustit server pro hru Counter-Strike se jako hodnota parametru zadá řetězec `cstrike`. Tento parametr je pro nás důležitý, potřebujeme jej znát pro správné určení cesty ke konfiguračním a dalším souborům serveru,

2 Herní engine - softwarový systém určený pro vývoj her, slouží hlavně k zjednodušení a urychlení práce při tvorbě počítačových her

podrobnější informace jsou uvedeny u parametru `+exec`. V případě nezadání se spustí Half-Life deathmatch server (hodnota: `valve`).

`-port <celé číslo v rozsahu 0 až 65535> (základní hodnota: 27015)`
Tento parametr slouží k zadání herního portu (port na který se připojují klienti). Tento port rovněž slouží jako dotazovací neboli *query* port (na tento port se zasílají požadavky pro zjištění stavu serveru, počtu hráčů, aktuálně běžící mapy, atp.). V případě, že není zadán, server nastaví hodnotu na `27015`.

`-sport <celé číslo v rozsahu 0 až 65535>`
Port sloužící ke komunikaci VAC2 (Valve AntiCheat 2 - software určený k odhalování a postihování podvodníků).

`+maxplayers <celé číslo v rozsahu 1 až 32> (základní hodnota: 6)`
Hodnota udává počet slotů neboli maximální počet hráčů, kteří mohou být souběžně připojeni k hernímu serveru.

`+map <řetězec>`
V případě nezadání tohoto parametru se na herní server nemohou připojit klienti, spuštění sice proběhne bez chyby, avšak server nenahraje žádnou mapu, na které by mohli klienti hrát. Bude tedy vhodné jej označit jako povinný.

`+ip <řetězec> (základní hodnota: 127.0.0.1)`
Při nezadání parametru server hlásí, že je spuštěn na adrese 127.0.0.1 (localhost), nicméně je dostupný na všech síťových rozhraních. Je vhodné jej nastavit pouze v případě, že chceme dostupnost herního serveru omezit na určitou adresu.

`-pingboost <celé číslo v rozsahu 1 až 3>`
Parametr ovlivňuje odezvu herního serveru. Z osobních zkušeností hodnota 1 nemá na odezvu velký vliv a při nastavení hodnoty 3 dosti rostou hardwarové nároky. Hodnota 2 je na úrovni mezi těmito dvěma, jeví se tedy jako nejvhodnější.

`+exec <řetězec>`
Hodnotou tohoto parametru je název konfiguračního souboru, jenž bude načten po spuštění serveru. Příkaz jednoduše projde soubor po řádcích a provede všechny příkazy v něm zadané.

Příklad příkazu pro spuštění (Counter-Strike)

```
./hlds_run -game cstrike -port 27015 +clientport 28015 -sport 29015
-pingboost 2 +maxplayers 12 +map de_dust2 +exec server.cfg
```

Nastavení serveru

Konfiguraci serveru provedeme nastavením jednotlivých CVAR[3] v konfiguračním souboru zadaném ve spouštěcích parametrech. V konfiguračním souboru se mohou vyskytovat komentáře. Komentářem je řetězec zapsaný na řádku za dvěmi lomítky `//` nebo znakem `#`. Syntaxe v konfiguračním souboru je shodná se syntaxí konzolových příkazů. Cesta ke konfiguračnímu souboru je `HOME/[game]/[exec]`.

3 CVAR – konfigurační proměnná (z angl. Configuration VARiable)

Syntaxe konzolových příkazů

```
hostname "Název herního serveru" //Nastavení názvu serveru
status //Výpis stavu serveru, počtu hráčů, aktuální mapy atd.
exec banned.cfg //Provedení příkazů z dalšího souboru
mp_timelimit 40 //limit jednoho kola v minutách
```

Hodnoty obsahující mezeru musí být obaleny do uvozovek tak, jako u CVAR `hostname` v ukázce výše.

3.1.2 Možnosti ovládání

GoldSource engine nabízí pouze jedinou možnost vzdálené správy. Skrze RCON protokol lze vzdáleně zasílat příkazy na konzoli herního serveru. RCON u GoldSource engine komunikuje skrze UDP protokol. Protože však v naší aplikaci budeme mít přímý přístup ke konzoli, nebudeme RCON potřebovat.

3.2 Source engine

V kapitole bylo čerpáno ze zdroje [6]. Jedná se o vylepšenou verzi GoldSource engine. Mezi hry postavené na Source engine patří:

- Half-Life 2 (2004)
- Counter-Strike: Source (2004)
- Day of Defeat: Source (2005)
- Portal (2007)
- Team Fortress 2 (2008)
- Left 4 Dead (2008)
- Left 4 Dead 2 (2009)

3.2.1 Spouštění a konfigurace herního serveru

Stejně jako u GoldSource engine je k dedikovanému serveru přiložen *shell* skript. Tento skript ošetřuje nastavení implicitních hodnot paramterů v případě jejich nenastavení skrze spouštěcí parametry.

Nejdůležitější spouštěcí parametry

`-game <řetězec> (základní hodnota: cstrike)`
Parametr `-game` určuje hru, pro kterou se má server spustit. Při požadavku spustit server pro hru Counter-Strike: Source se jako hodnota parametru zadá řetězec `cstrike`. Tento parametr je pro nás důležitý, potřebujeme jej znát pro správné určení cesty ke konfiguračním a dalším souborům serveru, podrobnější informace jsou uvedeny u parametru `+exec`. V případě nezadání se spustí Counter-Strike: Source server (hodnota: `cstrike`).

`-port <celé číslo v rozsahu 0 až 65535> (základní hodnota: 27015)`
Stejně jako u GoldSource engine slouží parametr k zadání herního portu (port na který se připojují klienti), rovněž slouží jako dotazovací neboli *query* port (na tento port se zasílají požadavky pro zjištění stavu serveru, počtu hráčů, aktuálně běžící mapy, atp.). V případě, že není zadán, server nastaví hodnotu na `27015`.

`+tv_port <celé číslo v rozsahu 0 až 65535> (základní hodnota: 27020)`
Součástí Counter-Strike: Source serveru je SourceTV (server sloužící k připojení diváků, kteří mohou sledovat hru). Hodnota určuje port, na který se tento SourceTV server namapuje.

`+maxplayers <celé číslo v rozsahu 1 až 32> (základní hodnota: 32)`
Hodnota udává počet slotů neboli maximální počet hráčů, kteří mohou být souběžně připojeni k hernímu serveru.

`+map <řetězec>`
V případě nezadání tohoto parametru se na herní server nemohou připojit klienti, spuštění sice proběhne bez chyby, avšak server nenahraje žádnou mapu, na které by mohli klienti hrát.

`-tickrate <celé číslo z množiny {33, 66, 100}>`
Možné hodnoty se mohou lišit u různých modifikací. Hodnotu 33, 66 a 100 lze nastavit u Counters-Strike: Source. Tento parametr udává frekvenci, s jakou server komunikuje s klienty a obnovuje veškeré informace o objektech ve hře. S vyšší hodnotou rostou požadavky na hardware.

`+ip <řetězec>`
Při nezadání parametru server hlásí, že je spuštěn na adrese 127.0.0.1 (localhost), nicméně je dostupný na všech síťových rozhraních. Je vhodné jej nastavit pouze v případě, že chceme dostupnost herního serveru omezit na určitou adresu.

`+exec <řetězec>`
Hodnotou tohoto parametru je název konfiguračního souboru, jenž bude načten po spuštění serveru. Příkaz projde po řádcích soubor a provede všechny příkazy v něm zapsané.

Příklad příkazu pro spuštění (Counter-Strike: Source)

```
./srcds_run -game cstrike -port 27015 +clientport 28015 -steamport
29015 +tv_port 26015 +ip 127.0.0.1 -pingboost 2 +maxplayers 12 +map
de_dust2 +exec server.cfg -tickrate 100
```

Nastavení serveru

Konfiguraci serveru provedeme nastavením jednotlivých CVAR, stejně jako u GoldSource engine. Komentáře se také zapisují stejným stylem jako u GoldSource engine. Syntaxe v konfiguračním souboru je shodná se syntaxí konzolových příkazů. Cesta ke konfiguračnímu souboru je `HOME/[-game]/cfg/[-exec]`.

Syntaxe konzolových příkazů

```
hostname "Název herního serveru" #Nastavení názvu serveru
status #Výpis stavu serveru, počtu hráčů, aktuální mapy atd.
exec banned.cfg #Provedení příkazů z dalšího souboru
mp_timelimit 40 #limit jednoho kola v minutách
```

Hodnoty obsahující mezeru musí být obaleny do uvozovek tak, jako u CVAR `hostname` v ukázce výše.

3.2.2 Možnosti ovládání

Source engine stejně jako GoldSource umožňuje ovládání s pomocí RCON protokolu. Zde je RCON implemetnován nad TCP protokolem. Ovládání je možné provádět zasíláním příkazů na konzoli serveru, což nám plně postačí.

3.3 Id tech engine

Existuje několik verzí Id tech engine, implementaci serveru pro operační systém Linux u nejpopulárnějších her však zajišťuje pouze jedna osoba externě pracující pro společnosti vyvíjející tyto hry. Herní servery jednotlivých titulů se tedy téměř vůbec neliší. Na tomto engine jsou postaveny například hry

- Quake 3 Arena (1999),
- Medal of Honor: Alied Assault (2002),
- Medal of Honor: Spearhead (2003),
- Wolfenstein: Enemy Territory (2003),
- Call of Duty (2003),
- Call of Duty 2 (2005),
- Call of Duty 4: Modern Warfare (2007),
- Call of Duty: World at war (2008).

3.3.1 Spouštění a konfigurace herního serveru

Nejdůležitější spouštěcí parametry

`+set net_IP <řetězec> (základní hodnota: 127.0.0.1)`
Je vhodné jej nastavit pouze v případě, že chceme dostupnost herního serveru omezit na určitou adresu, jinak je server dostupný na všech rozhraních

`+set net_port <celé číslo v rozsahu 0 až 65535> (základní hodnota: 28960)`
Parametr určuje port, na který se připojují klienti, rovněž slouží jako dotazovací neboli *query* port (na tento port se zasílají požadavky pro zjištění stavu serveru, počtu hráčů, aktuálně běžící mapy, atp.). V případě, že není zadán, server nastaví hodnotu na `28960`.

`+set dedicated <celé číslo v rozsahu 1 až 2> (základní hodnota: 2)`
Hodnota `2` (INTERNET) povoluje zasílání informací na master server (umožňuje zobrazení herního serveru v oficiálním seznamu serverů). Opakem je hodnota `1` (LAN).

`+set fs_game <cesta>`
Určuje adresář, ze kterého jsou načítány soubory serveru. V základu jsou soubory načítány z adresáře `HOME/main`, pokud je `fs_game` nastaven, jsou data načítána navíc i z `HOME/[fs_game]`, v případě duplicitních souborů v obou adresářích se načte soubor z `HOME/[fs_game]`. V případě, že cesta obsahuje mezery, je nutné řetězec obalit do uvozovek.

`+set sv_maxclients <celé číslo v rozsahu 1 až 64> (základní hodnota: 12)`
Maximální počet hráčů souběžně připojených k serveru.

`+map <řetězec>`
Tímto parametrem se nastavuje startovací mapa, ta musí být udána, v opačném případě server nenastartuje (není vypsána žádná chybová hláška).

`+set fs_homepath <cesta>`
Určuje adresář pro dočasné a další soubory (např. Logy), které server během chodu vytváří.

`+exec <řetězec>`
Hodnotou parametru je název konfiguračního souboru, jenž bude načten po spuštění serveru. Soubor je procházen po řádcích a jednotlivé příkazy jsou vykonávány.

Příklad příkazu pro spuštění (Call of Duty 2)

```
./cod2_lnxded +set net_IP 127.0.0.1 +set net_port 27015 +set
dedicated 2 +set fs_game pam +set ui_maxclients 12 +set
sv_maxclients 12 +map mp_burgundy +set fs_homepath
/server/call_of_duty +exec auto_exec.cfg
```

Nastavení serveru

Konfiguraci serveru provedeme nastavením jednotlivých CVAR[4] v konfiguračním souboru zadaném ve spouštěcích parametrech. V konfiguračním souboru se mohou vyskytovat komentáře. Komentáře jsou uvozeny dvěmi lomítky `//`. Syntaxe v konfiguračním souboru je shodná se syntaxí konzolových příkazů.

Cesta ke konfiguračnímu souboru závisí na nastavení spouštěcích parametrů `+set fs_game` a `+exec`. Umístění souborů může být následující: `HOME/[fs_game]/[exec]` (v případě, že je `+set fs_game` nastaven) nebo `HOME/main/[exec]`. Pokud nastane taková situace, že existují oba soubory, je zvolen `HOME/[fs_game]/[exec]`.

Syntaxe konzolových příkazů

```
set sv_hostname "Název herního serveru" #Nastavení názvu serveru
set sv_allowdownload 1 #Povolí stahování dat ze serveru
exec weapons.cfg #Provedení příkazů z dalšího souboru
```

Hodnoty obsahující mezeru musí být obaleny do uvozovek tak, jako u CVAR `sv_hostname`.

3.3.2 Možnosti ovládání

Herní server je možné ovládat přes konzoli nebo přes RCON protokol.

3.4 Unreal engine

Unreal engine se postupem času vyvíjel, důležité vlastnosti herních serverů však zůstaly zachovány. Do seznamu her, které jsou založeny na Unreal engine můžeme zařadit tyto:

- Unreal Tournament (1999)
- America´s Army (2002)
- Unreal Tournament 2004 (2004)
- Killing Floor (2005)

4 CVAR – konfigurační proměnná (z angl. Configuration VARiable)

- Red Orchestra: Ostfront 41-45 (2006)

3.4.1 Spouštění a konfigurace herního serveru

Nejdůležitější spouštěcí parametry

Informace v podkapitole byly čerpány z [7]. Spoutěcí parametry pro Unreal engine se od ostatních her dosti liší, obsahuje totiž jeden parametr, jehož formát je podobný formátu url adresy. V tomto parametru může být nastaveno hned několik hodnot různých proměnných.

```
server CTF-Face?game=Botpack.CTFGame?mutator=BotPack.NoRedeemer.rom?
game=KFmod.KFGameType?VACSecured=true?MaxPlayers=12
```

Právě tento parametr je specifický pro Unreal engine. Nebude vhodné jej nijak rozkládat na podpříkazy, existuje totiž mnoho nastavení, které se zde mohou vyskytnout a tato nastavení se liší u jednotlivých titulů postavených na tomto engine.

`ini=<řetězec>`
Hodnotou parametru je umístění konfiguračního souboru, ten musí být umístěn v adresáři `HOME/System/`.

`Log=<řetězec>`
Hodnotou parametru je umístění souboru určeného k zápisu logů, ten musí být umístěn v adresáři `HOME/System/`.

Příklad příkazu pro spuštění (Killing Floor)

```
./ucc-bin server CTF-Face?game=Botpack.CTFGame?
mutator=BotPack.NoRedeemer.rom?game=KFmod.KFGameType?
VACSecured=true?MaxPlayers=12 ini=UT.ini log=UT.log
```

Nastavení serveru

Nastavení serveru je uloženo v konfiguračním souboru, jenž je zadán spouštěcím parametrem `ini`. V tomto souboru jsou jednotlivá nastavení strukturována do bloků. Jednotlivé bloky začínají návěštím. Těchto návěští se v souboru objevuje několik desítek. Jejich název je uveden v hranatých závorkách (např. `[Engine.GameReplicationInfo]`). Tuto strukturu je třeba zachovávat, jinak nebudou změny aplikovány. V souboru není možné psát komentáře.

Syntaxe nastavení CVAR v konfiguračním souboru

```
ServerName=KF server
Port=27304
bEnabled=True
```

3.4.2 Možnosti ovládání

Herní servery založené na Unreal engine je možné ovládat pouze pomocí webového rozhraní, které je součástí serveru.

3.5 TrackMania Forever engine

Informace v celé kapitole jsou čerpány z [8]. TrackMania Forever engine na rozdíl od výše zmíněných není určen pro akční hry, ale pro závodní automobilové hry. Jedná se o vylepšený TrackMania engine. Na tomto engine jsou postaveny hry

- TrackMania Nations Forever (2008, neplacená verze),
- TrackMania United Forever (2008, placená verze).

3.5.1 Spouštění a konfigurace herního serveru

Spuštění serveru pro hry TrackMania vyžaduje autorizaci na master serveru. Autentizace se provádí ověřením přihlašovacího jména, hesla a validačního klíče. Způsob vytváření přihlašovacího účtu se u obou her liší (Nations verze používá pro klienta i server stejný způsob autorizace, serverový účet se vytváří v klientovi, pro United verzi existuje webová administrace, kde lze serverové účty přidávat a spravovat), využívají však stejný autorizační server. Server je u obou her až na drobné restrikce v případě Nations verze totožný. Tyto restrikce jsou určeny nastavením příslušných hodnot v konfiguračním souboru.

Nejdůležitější spouštěcí parametry

`/nodaemon`
Parametr zajistí, že server nebude spuštěn jako démon v pozadí.

`/dedicated_cfg=<řetězec>`
Jako řetězec se dosazuje název konfiguračního souboru. Hodnota musí být zadána, jinak se server nespustí. Pro konfigurační soubor je předpřipravena šablona dedicated_cfg.txt.

`/game_settings=<řetězec>`
Hodnotou je cesta k sekundárnímu konfiguračnímu souboru. V tomto případě se jedná o soubor s nastavením týkajícím se herních pravidel (časový limit mapy, seznam rotace map, atd.). V základní instalaci je předpřipraveno několik těchto souborů pro rozdílné obtížnosti map. Cesta se zadává relativně vzhledem k adresáři HOME/GameData/, jenž obsahuje veškerá data herního serveru. Hodnotou může být například MatchSettings/Nations/NationsRed.txt.

Příklad příkazu pro spuštění (Trackmania Nations Forever)

```
./TrackManiaServer /dedicated_cfg=dedicated_cfg.txt
/game_settings=MatchSettings/Nations/NationsRed.txt /nodaemon
```

Nastavení serveru

Konfigurační soubory této hry jsou v XML formátu. Spuštění serveru vyžaduje korektní nastavení autentizačních údajů a síťových parametrů. Ze souboru byly vyňaty pouze důležité elementy. Cesta ke konfiguračnímu souboru je `HOME/GameData/Config/[/dedicated_cfg]`.

Nastavení přihlašovacích údajů pro XML-RPC

```
<authorization_levels>
     <level>
          <name>SuperAdmin</name>
          <password>heslo/password>
```

```
      </level>
</authorization_levels>
```

Nastavení síťových parametrů

```
<system_config>
      <force_ip_address>127.0.0.1</force_ip_address>
      <server_port>27383</server_port>
      <server_p2p_port>28383</server_p2p_port>
      <bind_ip_address>127.0.0.1</bind_ip_address>
      <xmlrpc_port>26383</xmlrpc_port>
      <xmlrpc_allowremote>True</xmlrpc_allowremote>
      <packmask>stadium</packmask>
</system_config>
```

Nastavení přihlašovacího účtu na master server

```
<masterserver_account>
      <login>fhaccount27383</login>
      <password>ghjpass2</password>
      <validation_key>34F</validation_key>
</masterserver_account>
```

Nastavení ostatních důležitých údajů

```
<server_options>
      <name>Nazev serveru</name>
      <max_players>20</max_players>
      <password>heslo</password>
</server_options>
```

3.5.2 Možnosti ovládání

TrackMania servery je možné ovládat pouze skrze XML-RPC protokol.

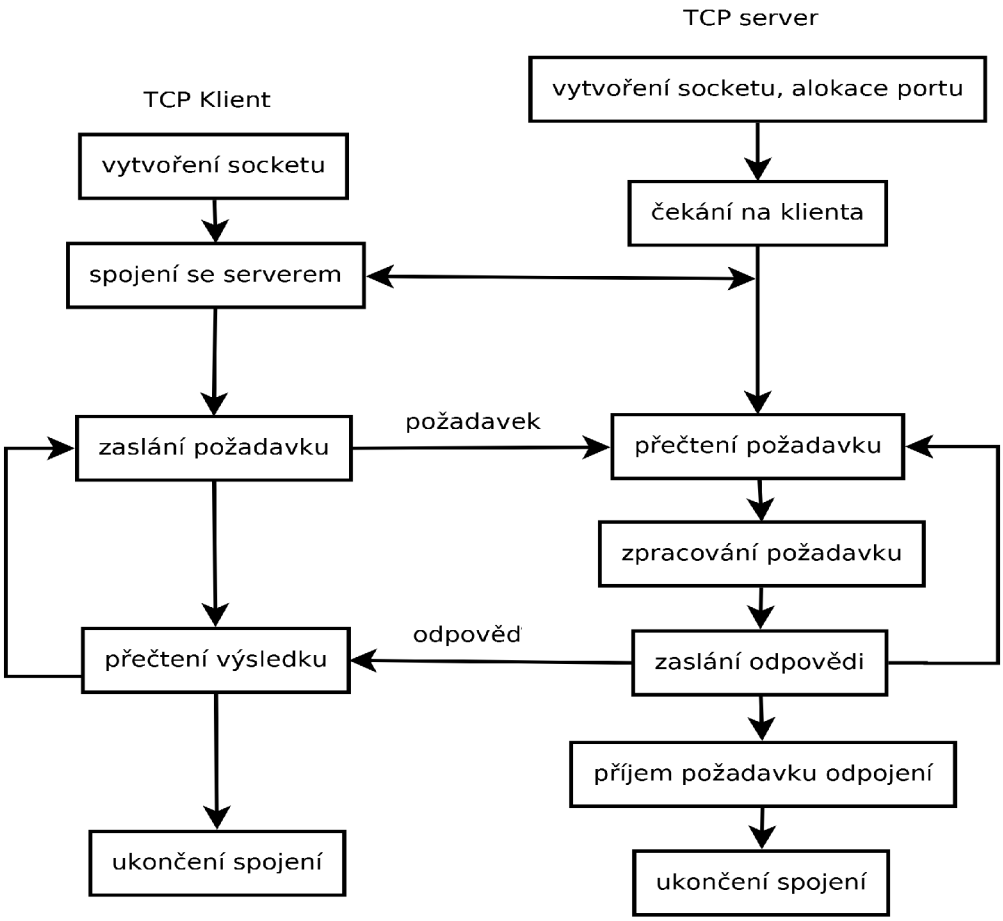
4 Návrh a implementace

Pro implementaci prototypu tohoto systému jsem si zvolil objektově orientovaný skriptovací jazyk Ruby a to hlavně díky jeho značným vyjadřovacím schopnostem. Při návrhu a implementaci byly čerpány informace ze zdroje [9].

4.1 Návrh komunikačního protokolu

Návrh komunikačního protokolu je velice důležitý. Musíme brát na vědomí, že s rozšířením o nový typ hry může přijít požadavek na implementaci nové funkce do systému. Dopředu také nevíme, jaká data bude potřeba zasílat, protokol by tedy měl být maximálně flexibilní.

Jako inspiraci jsem si zvolil XML-RPC protokol, jedná se o protokol, s jehož pomocí lze velice jednoduše provádět vzdálené volání procedur. V protokolu XML-RPC jsou data zapouzdřena do značkovacího jazyka XML a přenášena pomocí protokolu HTTP. Tyto informace byly čerpányy ze zdroje [10]. Komunikace bude probíhat způsobem znázorněným na obrázku 4.1



Obrázek 4.1: Klient - server komunikace a zpracování požadavku

4.1.1 Požadavek klienta

```
<request>
      <method>command</method>
      <params>
            <param>status</param>
      </params>
      <info/>

      <!-- Datový segment -->
</request>
```

4.1.2 Odpověď serveru

Server zasílá odpověď ve formátu:

```
<response>
      <info type='success'/>

      <!-- Datový segment -->
</response>
```

v případě úspěchu nebo

```
<response>
      <info type='error'>
            <message>Náhodná chyba</message>
            <code>50</code>
      </info>
</response>
```

v případě neúspěchu. Elementy `message` a `code` obsahují příslušné chybové údaje.

4.1.3 Datový segment

Pokud jsou součástí požadavku či odpovědi nějaká data, je doplněn datový segment. Ten se skládá z elementu `<data> </data>`, v němž je obsažena jediná proměnná (element `<variable> </varioable>`). Element `<variable> </variable>` může být následujícího typu.

Datový typ fixnum

Do tohoto datového typu lze uložit pouze celé číslo, typ `fixnum` v Ruby zastáva datový typ `long int` známý z jiných programovacích jazyků. V případě, že je zadáno číslo mimo rozsah, převede se automaticky na datový typ `bignum`, který má rozsah neomezený (používání tohoto datového typu výrazně zvyšuje režii).

```
<variable type="fixnum">
      <value>1024</value>
</variable>
```

Datový typ float

Slouží k uložení a přenosu desetinných čísel.

```
<variable type="float">
     <value>15.26</value>
</variable>
```

Datový typ string

Využívá se pro přenos libovolných řetězců. S pomocí tohoto typu se také přenáší obsah souborů, kódovaný algoritmem Base64.

```
<variable type="string">
     <value>Nějaký textový řetězec</value>
</variable>
```

Datový typ array

Slouží k uložení a přenosu polí. Prvky pole mohou být také pole, či asociativní pole, tímto je dosaženo uložení libovolného množství dat. Následující příklad reprezentuje pole s hodnotami [1024, "Nějaký textový řetězec"].

```
<variable type="array">
     <value index='0'>
          <variable type="fixnum">
               <value>1024</value>
          </variable>
     </value>
     <value index='1'>
          <variable type="string">
               <value>Nějaký textový řetězec</value>
          </variable>
     </value>
</variable>
```

Datový typ hash

Tento datový typ slouží k přenosu asociativních polí. Asociativní pole umožňuje libovolný počet zanoření stejně jako je tomu u běžného pole. Jediným rozdílem oproti poli je náhrada celočíselných indexů za řetězce (klíče). Následující příklad reprezentuje asociativní pole {"name" => "Nějaký text", "value" => 53.159}.

```
<variable type="hash">
     <value key="name">
          <variable type="string">
               <value>Nějaký text</value>
          </variable>
     </value>
     <value key="value">
          <variable type="double">
               <value>53.159</value>
          </variable>
     </value>
</variable>
```

4.2 Démon (GSDaemon)

4.2.1 Spouštění aplikace

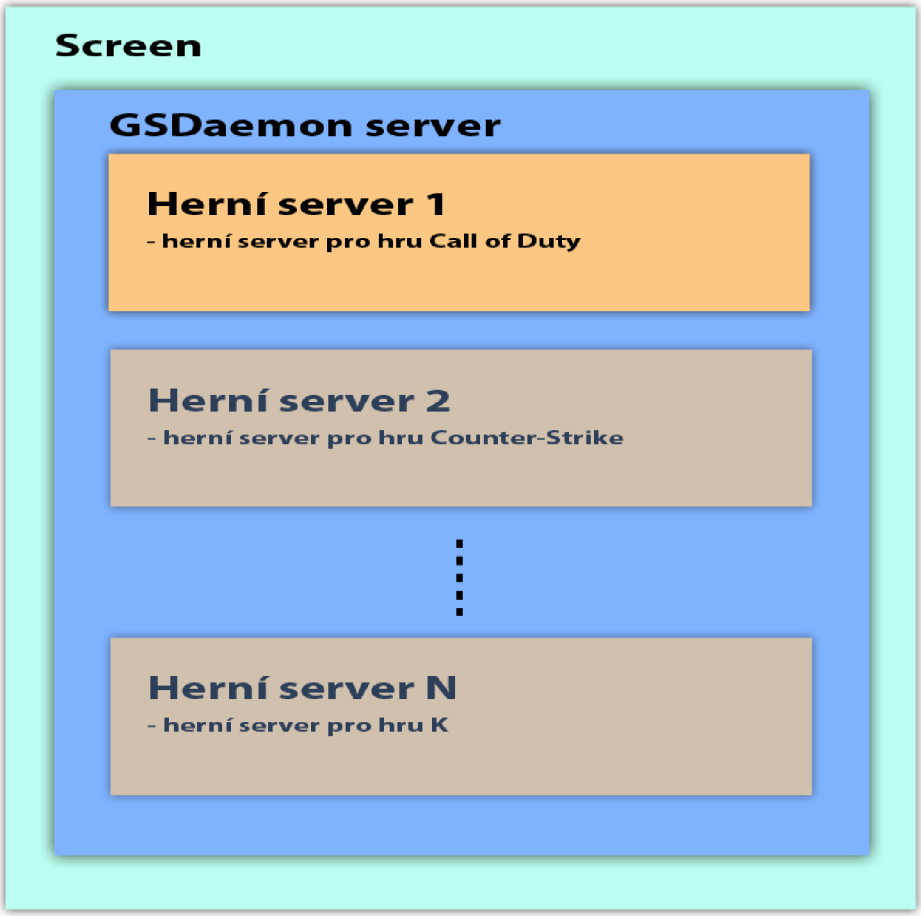
Struktura aplikace znázorněná na obrázku 4.2 vychází z požadavků na možnost správy lokální i vzdálené. Lokální správa bude prováděna utilitou Screen, skrze kterou bude spuštěn náš démon. Při vyvolání terminálu je vstup z naší klávesnice předáván na vstup démonu. Příkazy zadané s prefixem gsd_ démon odchytí, jsou to příkazy určené k jeho ovládání. Ostatní příkazy démon zasílá na vstup procesu herního serveru. V případě, že se nacházíme ve virtuálním terminálu, kombinace kláves CTRL + ALT + D zajistí odpojení, CTRL + C ukončí běh aplikace. Jak je z obrázku 4.2 patrné, GSDaemon umožňuje přepínání mezi více instancemi herních serverů, jenž mohou být i různého druhu.

Spuštění démonu utilitou Screen

```
screen -A -m -d -S gsdaemon_1 ./GSDaemon_run.sh
```

Vyvolání virtuálního terminálu démonu

```
screen -r  gsdaemon_1
```



Obrázek 4.2: Struktura aplikace (GSDaemon)

4.2.2 Konfigurace démonu

Po spuštění démonu je nejprve načtena konfigurace démonu samotného, tu nalezneme v konfiguračním souboru `config.xml`, tento soubor jak už přípona napovídá, je formátován značkovacím jazykem XML. Načtení konfigurace realizuje metoda *load* třídy *Configuration*. Metoda nejprve zparsuje soubor `config.xml`, V případě úspěchu z tohoto načteného nastavení zjistí název konfiguračního souboru daný elementem `ServersFile` s nastavením instancí herních serverů, ten také zparsuje a načte jejich nastavení. Pro získávání hodnot nastavení existuje několik metod, tyto metody není nutné popisovat, jelikož jsou uvedeny v diagramu tříd (podkapitola 4.2.10).

Příklad souboru config.xml

```
<GSDaemon>
      <Ip>217.11.249.85</Ip>
      <Port>29015</Port>
      <Login>admin</Login>
      <Password>admin</Password>
      <ConsoleBufferSize>64</ConsoleBufferSize>
      <ConsoleClientTimeout>60</ConsoleClientTimeout>
      <TransfersEnabled>true</TransfersEnabled>
```

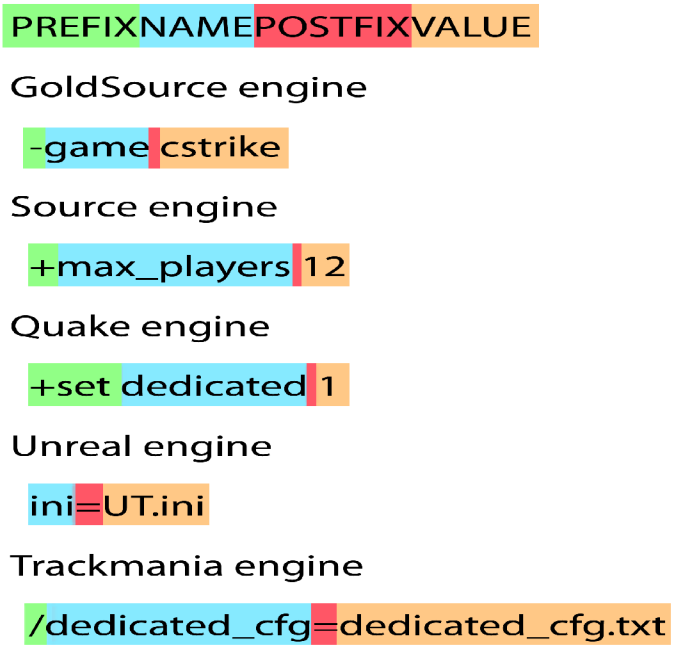
```
    <ServersFile>servers.xml</ServersFile>
    <DefaultServerId>04</DefaultServerId>
    <LogMaxFileSize>1048576</LogMaxFileSize> <!-- kB -->
    <StandardLogFile>std_#DATE#_#NUMBER#.log</StandardLogFile>
    <StandardLogEnabled>true</StandardLogEnabled>
    <ErrorLogFile>error_#DATE#_#NUMBER#.log</ErrorLogFile>
    <ErrorLogEnabled>true</ErrorLogEnabled>
    <DebugLogFile>debug_#DATE#_#NUMBER#.log</DebugLogFile>
    <DebugLogEnabled>false</DebugLogEnabled>
</GSDaemon>
```

Příklad souboru servers.xml

```
<Servers>
    <Server id="01">
        <GameFolder/>
        <GameExec>hlds_run</GameExec>
        <GameType>HalfLifeOld</GameType>
        <GameName>Counter-Strike 1.6</GameName>
        <GameShortcut>cs16</GameShortcut>
        <ServerName>Testovaci server CS 1.6</ServerName>
        <AutoRestart>true</AutoRestart>
        <DailyRestart>true</DailyRestart>
        <DailyRestartHour>4</DailyRestartHour>
        <!-- segment se spouštěcími parametry -->
    </Server>
</Servers>
```

4.2.3 Spouštěcí parametry herního serveru

Aby bylo možné přistupovat ke spouštěcím parametrům, libovolně je měnit, přidávat a odebírat, bylo třeba navrhnout univerzální strukturu pro jejich uložení. Po podrobnější analýze spouštěcích parametrů jednotlivých typů herních serverů jsem zvolil rozdělení na PREFIX-NÁZEV-POSTFIX-HODNOTA. Ukázkové rozdělení pro jednotlivé typy her je znázorněno na obrázku (obr.4.1).
Vzhledem k tomu, že jsou některé spouštěcí parametry povinné (herní server bez nich nenastartuje) a některé musí být zadány v určitém pořadí, byly do struktury přidány další dva údaje. Příznak `order`, který určuje v jakém pořadí budou parametry zapsány. Příznak `indelible` určující, zda je daný parametr povinný a není možné jej odstranit.



Obrázek 4.3: Názorné rozdělení spouštěcích parametrů

Pro spouštěcí příkaz

```
./hlds_run -game cstrike +ip 127.0.0.1 -port 27015 +map de_dust
+exec server.cfg
```

vypadá blok se spouštěcími parametry následovně:

```
<StartupParameters>
      <Parameter order="1" indelible="true">
            <Prefix>-</Prefix>
            <Name>game</Name>
            <Postfix> </Postfix>
            <Value>cstrike</Value>
      </Parameter>
      <Parameter order="2" indelible="true">
            <Prefix>+</Prefix>
            <Name>ip</Name>
            <Postfix> </Postfix>
            <Value>127.0.0.1</Value>
      </Parameter>

      <!-- Parametry 3 a 4 -->

      <Parameter order="5">
            <Prefix>+</Prefix>
            <Name>exec</Name>
            <Postfix> </Postfix>
            <Value>server.cfg</Value>
      </Parameter>
</StartupParameters>
```

4.2.4 Načtení pluginů pro jednotlivé typy her

Jakmile je nahrána konfigurace, přechází démon k načtení pluginu spouštěného herního serveru. Pluginy se vkládaji do adresáře `plugins`. Nejprve démon načte jejich seznam, zkontroluje, zda je v seznamu obsažen plugin pro spouštěný server (výchozí server je dán elementem `DefaultServerId`, obsaženým v konfiguraci démonu) a v případě, že tento plugin existuje, načte jeho obsah. Tento způsob zamezuje zbytečnému načítání pluginů, které nejsou zapotřebí. Pokud démon dostane žádost o přepnutí na jinou instanci herního serveru a požadovaný herní server je jiného druhu, načte se plugin pro tento druh dodatečně.

4.2.5 Spouštění procesu v operačním systému Linux

Dalším krokem po načtení pluginu dané hry, je spuštění instance herního serveru. Toto spuštění je realizováno metodou *start* třídy *GameServer*. Veškeré pluginy tuto metodu z třídy *GameServer* dědí. Programovací jazyk Ruby nabízí hned několik možností spouštění procesů. Nejvhodnější se nakonec ukázalo využití knihovny *PTY,* jedná se o standardní knihovnu, jenž implementuje možnost využití tzv. pseudo-terminálů v operačním systému Linux. Stejného rozhraní využívá utilita Screen popsaná v kapitole 2. Spuštění herního serveru provede metoda *spawn*, jediným parametrem metody je příkaz určený ke spuštění. Metoda vrátí tři proměnné, rozhraní pro zápis (*writer*), čtení (*reader*), tyto proměnné jsou typu *File* (pracuje se s nimi jako s otevřeným souborem) a poslední z těchto tří je PID (id nově vytvořeného procesu). Při volání metody *spawn* musí být pracovní adresář nastaven na adresář obsahující soubory herního serveru. Bohužel se při použití knihovny *PTY* vyskytl problém s odchycením výjimky, jenž má být vyvolána při ukončení spuštěného procesu. Tento problém jsem nakonec vyřešil implementací tzv. hlídacího psa (z angl. WatchDog), jedná se o metodu kontrolující existenci procesu s daným PID. Pokud hlídací pes zjistí, že proces herního serveru neběží, opětovně jej spustí.

4.2.6 Obsluha výstupu herního serveru

Zasílání příkazu procesu herního serveru je poměrně jednoduché. Stačí tento příkaz zaslat pomocí metody *write* nebo *puts*, jenž jsou dostupné pro instanční proměnnou *writer*. S čtením výstupu procesu už to tak jednoduché není. Pro práci se standardním výstupem procesu je připravena třída *Console*. Třída může mít v jeden okamžik vytvořenu pouze jednu instanci. Před vytvořením nové instance je třeba nejprve zavolat třídní metodu *Console.destroy,* tato zajistí zrušení původní instance. Jediným parametrem konstruktoru této třídy je výstupní datový proud (instanční proměnná *reader* třídy *GameServer* nebo jejích potomků). Z tohoto datového proudu je v cyklu po řádcích čten výstup procesu herního serveru, tento cyklus je spuštěn v novém vlákně. Třída si uchovává seznam tzv. čtenářů v třídní proměnné *readers*, tato proměnná je reprezentována polem instancí třídy *ConsoleReader*.

Instance třídy *ConsoleReader* jsou identifikovány unikátním identifikačním číslem (dále jen CUID). Tyto instance se dělí na dvě skupiny, takové jenž mají CUID v rozsahu od 1 do 99 (lokální čtenář) a ty s CUID vyšším než 99 (vzdálený čtenář). Lokálním čtenářem je myšlen takový u kterého je možno zapisovat přímo do datového proudu. Příkladem takového čtenáře je například standardní výstup. Vzdálený čtenář je takový, který se k démonu připojuje skrze síť a přečte vždy určitý počet řádků. Z těchto informací vyplývá, že u vzdáleného čtenáře je třeba uchovávat buffer s výstupem herního serveru.

Instance třídy *Console* si uchovává svůj vlastní buffer. Velikost tohoto bufferu se nastavuje konfiguračním elementem `ConsoleBufferSize`. Hodnotou tohoto elementu je počet uchovávaných řádků. Tento buffer umožňuje zaslat vzdálenému klientovi výstup z konzole už při jeho prvním požadavku (při prvním požadavku má čtenář prázdný buffer). Takovýto buffer se také hodí při ladění nastavení herního serveru, či jeho zacyklení. Při tomto konceptu zasílání výstupu

vzdáleným klientům bylo nutné ošetřit možné zaplnění paměti obrovským klientským bufferem. Z tohoto důvodu je po zápisu tolika řádek, jak je velký buffer, kontrolováno, jak dlouhá doba uběhla od posledního čtení vzdáleného čtenáře. Pokud tato doba přesáhne hodnotu konfiguračního elementu `ConsoleClientTimeout` zadaného v sekundách, je tento čtenář ze seznamu čtenářů odstraněn a jeho buffer je smazán. Čtení konzole vzdáleného klienta zprostředkovává metoda *console_read(cuid, lines)* třídy *GameServer*, jenž pouze předává výsledek třídní metody *Console.read_client_buffer(cuid, lines).*

4.2.7 Síťový server

Po úspěšném spuštění herního serveru je třeba inicializovat síťový server obsluhující klientské požadavky. Veškerou komunikaci s klientem zajišťuje instance třídy *Server.* Parametrem konstruktoru je instance třídy *GameServer* nebo jejích potomků. Ostatní údaje jako IP adresa či port, na kterém má být server dostupný, se načtou z konfiguračního souboru. Server je konkurentní – může obsluhovat více požadavků najednou. Jakmile se připojí klient, server mu vygeneruje náhodný identifikátor, který je uváděn v log souborech. Ze spojení je vytvořena nová instance třídy *client_session = ClientSession.new(session),* ta je jako parametr předána metodě *process_client*. Metoda *process_client* pouze v cyklu volá metodu *client_session.get_request,* poté se pomocí metody *client_session.quit?* zjistí, zda byl zaslán požadavek o ukončení spojení. V takovém případě cyklus ukončí, jinak volá metodu *handle_request(client_session)*, která provede požadovanou operaci, zašle odpověď a vrací se zpět do metody *process_client,* ta čeká na další požadavek.

Organizace dostupných metod

To, která metoda bude pro vzdáleného klienta dostupná a která ne určuje seznam dostupných metod uložený v třídní proměnné třídy *MethodManager. T*ato třída obsahuje pouze třídní metody. Metoda *MethodManager.add* obstarává přidání nové metody do seznamu dostupných metod. Metody jsou vkládány do kategorií podle toho, na jakou třídu se metoda váže. Veškeré metody, které mají být klientovi dostupné, musí být takto přidány. Podrobnosti o této metodě jsou uvedeny v podkapitole 4.2.9. Metoda *MethodManager.available?(method_name)* je volána při obdržení požadavku od klienta a rozhoduje, zda je možné požadavek obsloužit (to je možné v případě, kdy je metoda v seznamu). Další důležitou metodou je *MethodManager.delete_category(category)*. Tato metoda je volána při přepnutí na jinou instanci herního serveru, kdy je třeba z dostupných metod vymazat ty, jenž nebudou pro jinou instanci herního serveru dostupné.

4.2.8 Logování

Logování provozu a chyb, které se mohou vyskytnout v našem démonu, zajišťuje modul `Logger`. Tento modul obsahuje čtyři veřejné metody, z toho 3 určené k zápisu: *debug, info, error* a jednu pro vrácení poslední chyby: *get_last_error*. Zápisové metody mají dva parametry, prvním je zpráva a druhým je typ zápisu, nabývá hodnot: 1 - zápis na standardní výstup / chybový výstup , 2 - zápis na standardní výstup / chybový výstup a do souboru, 3 - zápis do souboru. Druhý parametr má implicitní hodnotu nastavenu na 2. Logy se zapisují do adresáře `logs` do souborů, jejichž název se odvíjí od vzoru zadaného v konfiguračním souboru démonu. Do tohoto vzoru je možné zadat požadovaný název souboru pro jednotlivé metody. V tomto vzoru dojde k náhradě #DATE# za datum ve formátu YYYY-MM-DD a #NUMBER# za pořadové číslo logu. V případě, že nejsou tyto zástupné řetězce v konfiguračním souboru zadány, připojí se za nastavený vzor #DATE#_#NUMBER#. Privátní metoda *get_real_filepath* zajišťuje rotaci log souborů podle data a velikosti souboru, která je omezena konfiguračním elementem `LogMaxFileSize`.

4.2.9 Způsob rozšíření o další typ hry

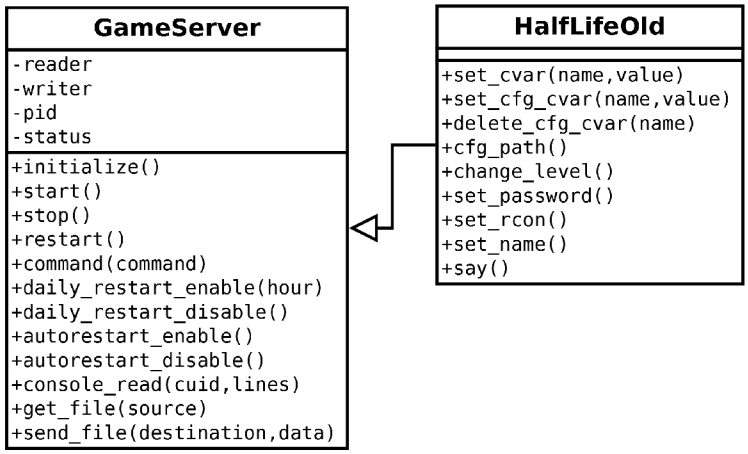
Rozšíření démonu o nový typ hry se provádí vytvořením nového pluginu. Prakticky jde o vytvoření nové třídy, která dědí od třídy *GameServer* viz obrázek 4.4. Takto vytvořená třída se uloží do souboru pod názvem, jenž je odvozen od názvu třídy tak, že všechna velká písmena kromě prvního jsou nahrazena za písmeno malé následované podtržítkem a doplněna koncovkou .rb (např. třída *HalfLifeOld* => `half_life_old.rb`). Při psaní pluginu je možné využívat metod děděné třídy a taky všech třídních metod tříd: *Configuration, Logger* a *MethodManager*. Veškeré metody, které chceme zpřístupnit, musí být přidány do seznamu dostupných metod pomocí *MethodManager.add*, jak je demonstrováno níže.

Demonstrace vytvoření nového pluginu

Obsah souboru half_life_old.rb

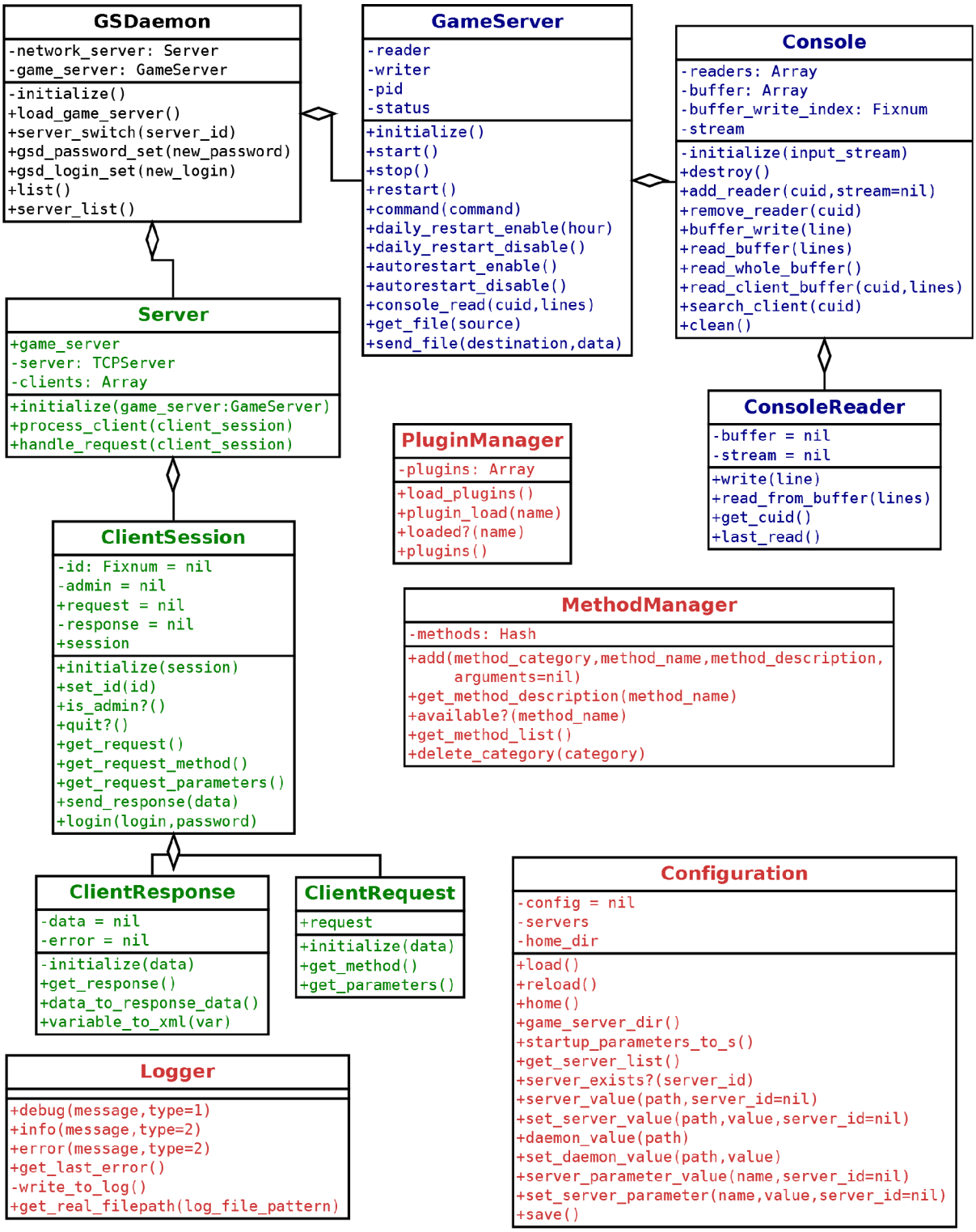
```
class HalfLifeOld < GameServer

  METHOD_CATEGORY = 'HalfLifeOld'
  MethodManager.add(METHOD_CATEGORY,'say', 'Vypise <zpravu> hracum
na serveru', [['zprava', '']])
  MethodManager.add(METHOD_CATEGORY,'set_cvar', 'Nastavi <cvar> na
danou <hodnotu>', [['cvar', ''], ['hodnota', '']])

  def set_cvar(name, value)
    return self.command("#{name} \"#{value}\"")
  end

  def say(message)
    return self.set_cvar("say", message)
  end
end
```

V tomto příkladu jsou pro názornost implementovány pouze dvě metody z diagramu tříd na obrázku 4.4, u dalších metod by se postupovalo podobně.



Obrázek 4.4: GSDaemon - rozšíření, dědičnost třídy

4.2.10 Diagram tříd



Obrázek 4.5: GSDaemon - diagram tříd

4.3 Klient (GSClient)

Jedná se o jednoduchého konzolového klienta využívajícího knihovny `g_s_client.rb`. V té je obsažena implementace tříd *GSClient*, *GSRequest* a *GSResponse*. Po spuštění jsou rozparsovány spouštěcí parametry, které mohou být zadány v libovolném pořadí. Jediný parametr `-m` musí být až na posledním místě. Následuje vytvoření instance *client = GSClient.new(host, port, user, password)* a je zavolána metoda *send_request,* jejímž prvním parametrem je název metody, druhým data a třetím pole případných argumentů. Tato metoda vrací instanci třídy *GSResponse*, úspěšnost požadavku lze ověřit zavoláním metody *success?* na tuto instanci. Pokud nastala nějaká chyba, lze ji vypsat metodou *read_error*. V případě, že jsou součástí odpovědi nějaká data, jsou uložená ve veřejné instanční proměnné *data*.

Použití knihovny g_s_client.rb - názorné zaslání požadavku a vypsání výsledku

```
client = GSClient.new("test.cz", 29015, "admin", "admin")
response = client.send_request("list")
puts response.read_error unless response.success?
puts response.data.inspect if response.success?
client.disconnect
```

4.3.1 Spouštěcí parametry

-h, --help	zobrazení nápovědy
-host	IP adresa nebo doménové jméno serveru
-port	port, na kterém běží GSDaemon
-l	přihlašovací jméno
-p	přihlašovací heslo
-m	název metody následovaný případnými argumenty, tento parametr musí být uveden až jako poslední

Všechny tyto spouštěcí parametry jsou povinné.

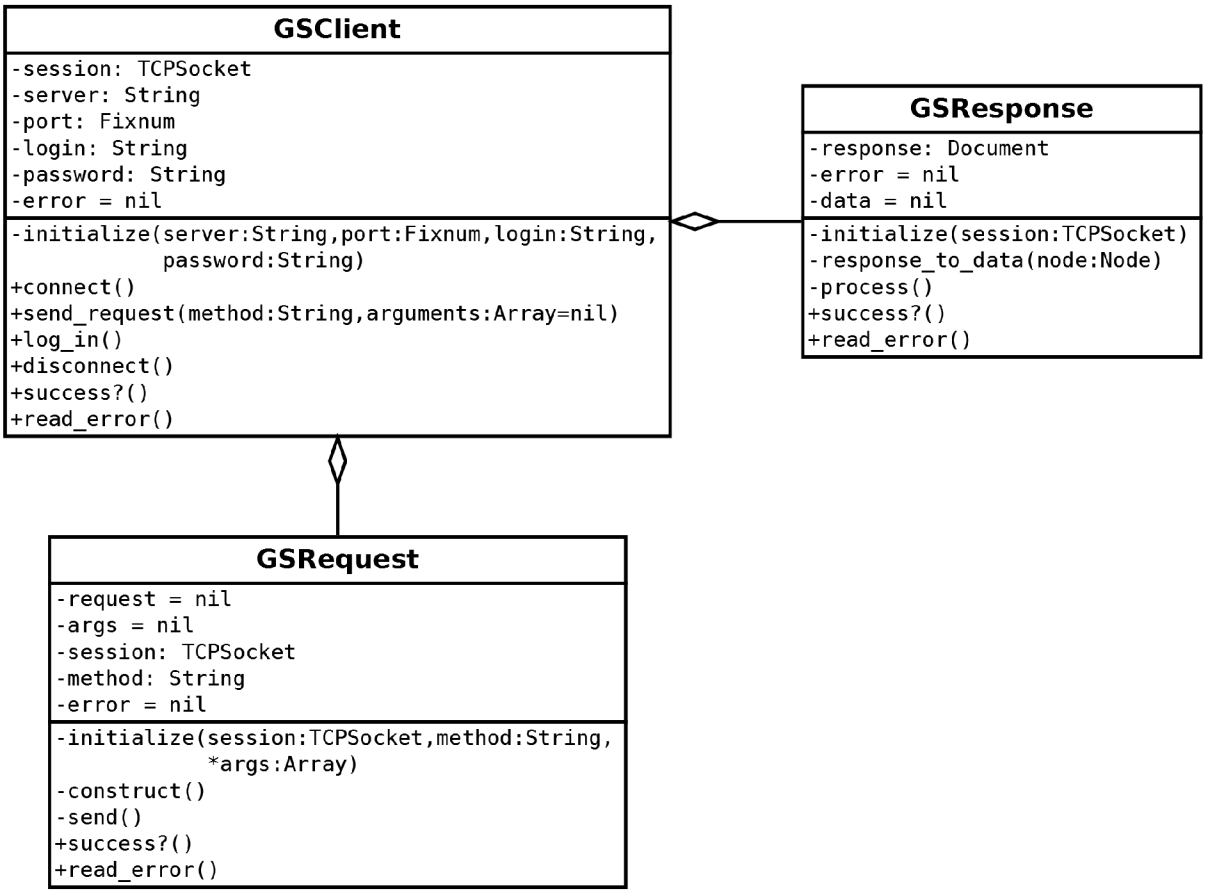
Příklad užití (výpis dostupných metod)

```
ruby GSClient.rb -host test.cz -port 29015 -l admin -p admin -m list
```

4.3.2 Způsob rozšíření o další typ hry

Vzhledem k tomu, že je klient navržen tak, aby pouze volal metody démonu, není třeba jej při požadavku o rozšíření démonu o novou hru nijak upravovat.

4.3.3 Diagram tříd



Obrázek 4.6: GSClient - diagram tříd

4.4 Zabezpečení

Tento systém je primárně vyvíjen za účelem použití s webovou aplikací [10]. Předpokládá se, že webový server bude umístěn ve stejném datacentru jako server s naším démonem, data by tedy měla proudit pouze po lokální síti, kde by nemělo hrozit jejich odposlouchávání. V případě, že by byla vytvořena klientská aplikace komunikující s našim démonem přes internet, bylo by vhodné komunikaci šifrovat. Bylo by to vhodné, né však nutné, protože většina herních serveru obsahuje síťová administrační rozhraní (RCON, XML-RPC), která také nejsou nijak šifrována. Jediným zabezpečením je pouze zašifrování přihlašovacího hesla.

5 Závěr

5.1 Další vývoj

Tento systém je navrhnut pro práci s webovým klientem, který řeší veškerou správu uživatelských účtů a jejich práva. Náš systém umožňuje pouze jednu úroveň oprávnění a to administrátor s plnými právy. V případě rozšíření systému o klientský software spouštěný na uživatelském počítači by bylo vhodné doimplementovat správu účtů do démonu. Dalším možným rozšířením je načítání konfigurace z databáze, vhodným konceptem by bylo načítání databáze → soubor → démon, který by zamezil výpadkům herního serveru v případě výpadku databáze. Po uvedení systému do finální stabilní verze by bylo vhodné jej přepsat do efektivnějšího programovacího jazyka, kupříkladu C++.

5.2 Zhodnocení

Tato práce mi přinesla mnoho nových zkušeností s programovacím jazykem Ruby. Zdokonalil jsem zde své schopnosti návrhu objektově orientovaných aplikací a dozvěděl se nové informace o herních serverech. Při implementaci systému jsem narazil na několik problémů, nejčastějším z nich byla zcela chybějící dokumentace některých herních serverů, v takových případech jsem čerpal z několika nepříliš věrohodných zdrojů a takto získané informace sám ověřoval testováním. Jsem přesvědčen, že mi tyto nově nabyté znalosti a dovednosti budou přínosem jak v budoucím studiu, tak i případné profesi.

Literatura

[1] Tomáš Cigán: *Klient pro správu herních serverů*, bakalářská práce, Brno, FIT VUT v Brně, 2010

[2] Kolektiv autorů: *Screen* – Manuál verze 4.1.0, 2003, Dostupný z WWW: <http://www.gnu.org/software/screen/manual/screen.pdf>

[3] WWW stránky: Počítačová hra. URL <http://cs.wikipedia.org/wiki/Počítačová_hra>

[4] WWW stránky: Battlefield Ranked Server ROE. URL <http://forums.gameservers.com/viewtopic.php?f=9&t=21584>

[5] WWW stránky: Game engine. URL <http://en.wikipedia.org/wiki/Game_engine>

[6] WWW stránky: Command Line Options. URL <http://developer.valvesoftware.com/wiki/Command_Line_Options>

[7] WWW stránky: Unreal Console Commands. URL <http://unreal.epicgames.com/UTConsole.htm>

[8] WWW stránky: TrackMania Forever. URL <http://www.gamers.org/tmf/>

[9] Hal Fulton: *The Ruby Way: Solutions and Techniques in Ruby Programming, Second Edition.* Addison Wesley Professional, 2006, ISBN 0-672-32083-5

[10] Simon St. Laurent, Joe Johnson, Edd Dumbill: *Programming Web Services with XML-RPC*. O'REILLY, 2001, ISBN 0-596-00119-3