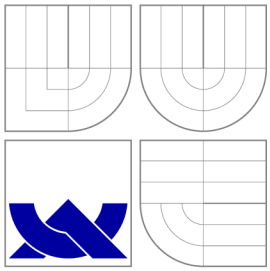
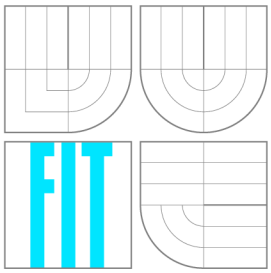


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# APLIKACE PRO ODESÍLÁNÍ SMS PŘES WEBOVÉ BRÁNY S VYUŽITÍM ESMSKA MODULŮ

AN APPLICATION UTILIZING ESMSKA PLUGINS FOR SENDING SMS MESSAGES VIA WEB GATEWAYS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DOMINIK MARTON

VEDOUCÍ PRÁCE
SUPERVISOR

RNDr. MAREK RYCHLÝ, Ph.D.

BRNO 2015

# Abstrakt

Tato práce popisuje návrh a implementaci konzolového programu, který pomocí JavaScript modulů programu esmska umožňuje z uživatelských skriptů odesílat SMS zprávy skrz webové brány bez CAPTCHA zabezpečení, a služby zpřístupňující program vzdáleně pomocí REST rozhraní. Program i služba jsou určeny především pro běh na síťových prvcích s operačním systémem OpenWrt, a proto se snaží minimalizovat paměťové nároky. Řešení je napsané kompletně v jazyce C s použitím knihovny libcurl, vestavěného JavaScript interpretu Duktape a standardní knihovny jazyka C uClibc.



# Abstract

This thesis describes the design and implementation of a console application, that utilizes esmska JavaScript modules to enable sending SMS messages from inside of user scripts via web gateways without CAPTCHA security plugin, and of a web service with REST interface making this application accessible remotely. Both are primarily made with the purpose of running on network devices with OpenWrt operating system while trying to minimise memory consumption. The whole solution is written in C with the help of libcurl library, embedded JavaScript interpreter Duktape and standard C library uClibc.



# Klíčová slova

SMS, JavaScript, Duktape, libcurl, C, OpenWrt, směrovač, síťový prvek, webová brána, CAPTCHA, Linux, esmska, uClibc

# Keywords

SMS, JavaScript, Duktape, libcurl, C, OpenWrt, router, network device, web gateway, CAPTCHA, Linux, esmska, uClibc


# Citace

Dominik Marton: Aplikace pro odesílání SMS přes webové brány s využitím esmska modulů, bakalářská práce, Brno, FIT VUT v Brně, 2015

# Aplikace pro odesílání SMS přes webové brány s využitím esmska modulů

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana RNDr. Marka Rychlého, Ph.D. a uvedl všechny prameny a publikace, ze kterých jsem čerpal.

.....
Dominik Marton
19. května 2015

## Poděkování

Děkuji vedoucímu této bakalářské práce, panu RNDr. Markovi Rychlému, Ph.D., za trpělivé odpovědi na mé dotazy, motivaci a podporu při jejím zpracovávání.

# Obsah

# Kapitola 1

# Úvod

V dnešní době si mnoho lidí nedovede život bez mobilního telefonu představit. Je možné člověka pohodlně a rychle kontaktovat v libovolnou denní dobu. Chytré telefony díky nejrůznějším aplikacím poskytují mnoho možností, o kterých se nám u prvních modelů přenosných telefonů ani nesnilo. Podobně je to i s internetovým připojením, na kterém je závislých mnoho velkých firem, poskytovatelů služeb a internetových burz. Fungování a stabilita internetového spojení jsou v mnoha případech kritické a jeho ztráta by mohla znamenat značný peněžní prodělek, a proto je monitorování a udržování sítí důležité. Přepracování esmska programu do konzolové podoby by vytvořilo způsob okamžitého informování v případě výskytu problému na jednom ze sledovaných síťových uzlů přímo na mobilní telefon.

Tato práce se podrobně věnuje vytváření zmiňované konzolové verze programu esmska. Kapitola 2 se zabývá prvotní analýzou problému, původní implementace v Javě a dostupných prostředků, technik a technologií vhodných pro účel výsledného programu. V kapitole 3 je uveden návrh struktury programu a zdůvodnění použití zvolených komponent a prostředků. Kapitola 4 popisuje implementaci a fungování jednotlivých částí programu a REST služby. Fakta zjištěné z profilování konzolového programu jsou krátce prezentovány v kapitole 5. Poslední kapitola 6 shrnuje a hodnotí výsledky práce.

# Kapitola 2

# Analýza

## 2.1 Operační systémy na síťových prvcích

Aktivní síťový prvek, v dnešní době především směrovač a přepínač, je specializovaný počítač určen ke zpracovávání paketů na různých vrstvách ISO/OSI modelu. Jedná se o základní součást síťové infrastruktury, s jejíž růstem se zvyšují i výkonnostní požadavky na tyto prvky. Nastal tedy stav, kdy je zapotřebí hardwarové prostředky jednotně spravovat pomocí operačního systému. Mezi hlavní představitele patří proprietarní IOS[1] (Internetworking operating system) od společnosti Cisco, používaný na Cisco zařízeních u velkých síťových a firemních uzlů, jenž je charakteristický různými pracovními módy příkazové řádky a sadou vlastních směrovacích a monitorovacích protokolů. Na opačné straně stojí volně dostupný a otevřený OpenWrt[2] z Unix rodiny, určený především pro vestavěná zařízení a domácí směrovače s omezenými hardwarovými prostředky. Je dostupný pro desítky různých architektur používající různé instrukční sady. Hlavní komponenty systému tvoří monolitické linuxové jádro, sada standardních a ořezaných verzí utilit Linuxových systémů a standardní knihovna jazyka C uClibc[3]. OpenWrt poskytuje také správce balíčků opkg[4], přes který je možno stáhnout programy a knihovny z oficiálního repozitáře této distribuce. Správu zařízení je možno vykonávat přes příkazovou řádku nebo přes webové rozhraní.

## 2.2 uClibc

Každý operační systém patřící do rodiny Unix vyžaduje pro svůj běh standardní knihovnu jazyka C, která obsahuje makra, definice datových typů, sady funkcí pro práci s řetězci, vstupem a výstupem, matematickými výpočty, a funkce pro zpřístupnění služeb operačního systému. Většina Linuxových distribucí je postavena na implementaci GNU C knihovny glibc[5]. Jelikož první verze byla napsána v roce 1987[1], nese si tato implementace s sebou značnou část historie, která může moderní programy zatěžovat. Vestavěná zařízení nabízí pouze omezené prostředky, kde je každá jednotka paměti i výkonu důležitá. Pro aplikaci v těchto zařízeních bývá glibc příliš nafouklá, protože zachovává zpětnou kompatibilitu pro starší programy, která je v tomto případě nepotřebná a zbytečná. Pro tento účel vzniklo

[1] http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-software-releases-listing.html

[2] https://openwrt.org

[3] http://www.uclibc.org

[4] https://code.google.com/p/opkg/

[5] http://www.gnu.org/s/libc/

několik alternativních implementací jako Bionic[6], užívaná operačním systémem Android[7], nebo uClibc, původně vytvořena pro uClinux[8], jenž je odnož Linuxového jádra pro mikrokontroléry bez MMU jednotky. uClibc se snaží poskytovat co nejvíce funkcí v co nejmenší velikosti knihovny. Zaobírá se pouze operačními systémy s jádrem Linux, což vede ke zjednodušení implementace a poskytnutí prostoru pro optimalizace. Podpora některých funkcí knihovny je ve výchozím stavu zakázána (např. plná podpora knihovny Math v souladu se standardem C99[9], protokol IPv6 nebo RPC), které mohou být dodatečně povoleny.

| **Porovnávaný objekt** | **glibc** | **uClibc** |
|---|---|---|
| Kompletní soubor statických knihoven | 2,0 MB | 500 kB |
| Kompletní soubor dynamických knihoven | 7,9 MB | 560 kB |
| Nejmenší statický program v jazyce C | 662 kB | 5 kB |
| Statický program "hello world"(použito printf) | 662 kB | 70 kB |
| Režie dynamického linkování | 48 kB | 40 kB |
| Režie statického linkování | 28 kB | 12 kB |
| Režie statického stdio | 36 kB | 24 kB |

Tabulka 2.1: Srovnání zátěže sekundární paměti standardními knihovnami

## 2.3 Optimalizace

Optimalizovat lze často pouze jeden aspekt programu, většinou na úkor jiného. Je tedy zapotřebí zvážit, který prostředek je vhodné upřednostnit – procesorový čas nebo paměť, ať již operační nebo sekundární. Tyto prostředky jdou principiálně proti sobě. Zvýšením paměťové náročnosti programu lze urychlit jeho běh, a naopak užitím vyššího výpočetního výkonu umožňuje snížit potřebnou paměť. V praxi se obvykle volí kompromis mezi těmito dvěmi doménami, jenž bývá aplikačně závislý. Např. v multimédiích, komprimovaná videa jsou vhodná pro streamování nebo pro tzv. video on demand služby, neboť komprese, ztrátová či bezztrátová, vede k menším datovým tokům. V případě úprav a práce s videi je upřednostňována rychlost zobrazení zpětné vazby po provedení změny. K provedení změny by tedy bylo potřeba video dekódovat, změnu aplikovat a znovu zakódovat, což je časově náročná operace, kvůli které dochází ke ztrátě kvality videa. Video je tedy uloženo bez komprese a pracuje se s ním pouze lokálně.

Výsledkem optimalizace nebývá optimální řešení pro zvolenou doménu, ale pouze její aproximace. Optimalizace lze realizovat na několika úrovních. Ty na nižších úrovních vyžadují mnoho práce a tvoří pouze malé procento celkového zisku. Čím výše v hierarchii postupujeme, tím zásadněji změny ovlivňují výsledný program, čímž roste i dopad na potřebné prostředky. Nejvyšší úroveň představuje návrh. Jsou známy požadavky a omezení výsledného programu, na základě nichž návrh staví a na které spoléhá. Při rozhodování se volí např. prostředky jako programovací jazyk nebo platforma. O úroveň níž se nachází algoritmy a datové struktury. Volba datových struktur je důležitá, neboť v pozdějších stádiích

[6] https://android.googlesource.com/platform/bionic.git
[7] https://www.android.com
[8] http://www.uclinux.org
[9] http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=29237

vývoje by jejich změna mohla představovat nutnost přepsání velkého kusu zdrojového kódu. Změna algoritmů je obecně jednodušší a při jejich výběru je vhodné se vyhýbat těm s polynomialní složitostí, časovou nebo prostorovou, podle zvolené optimalizační domény. Dále se nachází úroveň zdrojového kódu, kde záleží na způsobu zápisu požadovaného chování. Pro optimalizaci na této úrovni je zapotřebí velice dobrá znalost zvoleného programovacího jazyka a jeho překladače, aby autor kódu používal konstrukce, které se ve výsledku přeloží na efektivní soubor symbolických instrukcí. Na úrovni sestavování se při překladu povolují direktivy ovlivňující funkce programu, jenž budou v konečném spustitelném souboru, nebo které např. vyberou vhodnější implementaci některých částí pro cílovou platformu. Úroveň překladače představuje zapnutí optimalizací prováděných překladačem, obvykle ohledně velikosti paměťového otisku, rozvíjení cyklů aj. Nejnižší a nejtěžší je úroveň symbolických instrukcí. Psaní v jazyce symbolických instrukcí bývá vyžadováno pouze v případě vestavěných mikroprocesorových zařízení. Umožňuje efektivně využít celý repertoár dostupných instrukcí dané platformy. Jelikož jsou v dnešní době programy překládány tak, aby je bylo možné používat na co nejširším spektru strojů, je velmi těžké vytvořit efektivnější kód s identickou mírou přenositelnosti. Praktické řešení představuje přeložení zdrojového kódu z vyššího programovacího jazyka a v případě nutnosti následnou optimalizaci kritických sekcí, neboť psaní celého programu jen pomocí symbolických instrukcí je příliš nákladné.

## 2.4 Zabezpečení webových bran proti robotům

Umožnění zasílání SMS zpráv prostřednictvím internetu s sebou nese riziko snadného zneužití, nejpravděpodobněji na rozesílání spamu a reklam, popř. k provedení DDoS útoku na samotnou webovou bránu. Je tedy potřeba nasadit protiopatření a minimalizovat negativní dopad této služby. K odlišení člověka od robota se používají různé formy Turingova testu, nejčastěji s použitím obrázku. Mezi nejrozšířenější řešení patří CAPTCHA[10] webový plugin (zkratka pro Completely Automated Public Turing test to tell Computers and Humans Apart)[5]. Uživateli zobrazí vygenerovaný obrázek s textem, který je do jisté míry deformován, a předpokládá se, že člověk bude schopen i přesto deformovaný text rozpoznat a správně zapsat, zatímco metody optického rozpoznávání znaků (zk. OCR) neuspějí. Jelikož techniky OCR se stále progresivně vyvíjí, je potřeba k překonání těchto technik text více a více deformovat, a často již dochází k situaci, kdy ani člověk není schopen text jednoznačně rozluštit. Jedno z řešení je možnost strojového přečtení zdeformovaného textu v obrázku. Další implementace této ochrany představuje reCAPTCHA[11]. Rozdíl spočívá v sadě obrázků nabízených uživateli. Při převodu novinových článků a knih do digitální podoby se objevují části textu nebo texty na fotkách, které OCR software není schopen rozpoznat. Tyto pasáže spolu s již známým kontrolním slovem jsou předváděny respondentům a jejich odpovědi jsou zaznamenávány. Z odpovědí, které obsahovaly správné kontrolní slovo, je vybrána odpověď s určitým bodovým ohodnocením a ta je začleněna do výsledné digitální formy. Mezi další formy Turingova testu patří zobrazení otázky, typicky v anglickém jazyce, v obrázku. Otázka může být textového charakteru nebo ve formě jednoduchého matematického výrazu.

[10] http://www.captcha.net
[11] https://www.google.com/recaptcha

## 2.5 Architektura původního programu esmska

Originální implementace[12] je napsána v jazyce Java, díky čemuž je dosaženo nejvyšší úrovně přenositelnosti, kterou lze pomocí Java Virtual Machine (zk. JVM) zajistit. Hlavními nevýhodami této volby jsou však typické rysy programů napsaných v tomto jazyce, a to značné paměťové nároky a vzhledem k nenativní povaze jazyka také větší výkonnostní požadavky hardwaru. Program poskytuje grafické uživatelské rozhraní ke snadné obsluze. Mezi jeho hlavní funkce patří zpřístupnění odesílání SMS zpráv skrz webové brány, možnost ukládání přihlašovacích údajů k jednotlivým bránám, kontaktů do telefonního seznamu, import a export kontaktů, barevné přizpůsobení rozhraní a indikátor počtu napsaných znaků v aktuální rozepsané SMS zprávě. Pokud uživatel zvolí webovou bránu, u které není vyžadována registrace, při kliknutí na tlačítko zahajující proces odesílání je zobrazeno okno s bezpečnostním CAPTCHA obrázkem, ve kterém je uživatel vyzván k opsání textu na obrázku.

Přímá práce s webovými branami je přítomna v oddělených modulech, jenž jsou napsány v jazyce JavaScript. Každý modul se skládá ze sady funkcí vracející informace o dané webové bráně, např. maximální možnou délku zprávy, webovou adresu brány, její název a popis, nebo výčet vlastností brány, například, zda je potřeba se k bráně přihlásit, jestli k odeslání SMS zprávy je vyžadováno opsání CAPTCHA textu, nebo jestli brána poskytuje doručenku jako zpětnou vazbu. Nacházejí se v nich i údaje o samotném modulu, jako číslo jeho verze a datum vyhotovení, a kontakt na osobu, většinou ve formě emailové adresy, udržující daný modul. Klíčovou funkcí je však `send`, ve které se odehrává vlastní interakce s bránou a odesílání zprávy. Ta může volat další podpůrné podprogramy v rámci modulu.

Javovský řídící program a moduly však nejsou zcela odděleny. Prvním důvodem je hlášení chyby v případě výskytu problému při odesílání zprávy. Je za potřebí informovat hlavní program o výsledku činností modulu. Funkce `send` je navržena tak, že návratová hodnota pouze sděluje, zda byla SMS zpráva odeslána korektně nebo došlo k potížím. V případě, že by funkce vracela kromě dvoustavové proměnné také detailní popis nastalé chyby, např. ve formě objektu s pevně stanovenými atributy, byla by uvedená příčina bezpředmětná. Druhým důvodem je komplexnost potřebných operací. Jelikož se manipuluje s obsahem webové stránky, je nezbytné mít k dispozici příslušné funkce, které však přesahují rámec čistého JavaScriptu. Obě propojení, vzniklá ze zmiňovaných příčin, jsou dosažena pomocí funkcí v hlavní části programu, implementující komunikační prostředky a chybějící složitější funkce, které jsou posléze JavaScript modulům zpřístupněny a zevnitř funkce `send` volány. Jádro programu tvoří pouze pár tříd, ze kterých právě ve třídě `GatewayExecutor` jsou přítomny rozšiřující funkce.

## 2.6 SOAP a REST webové služby

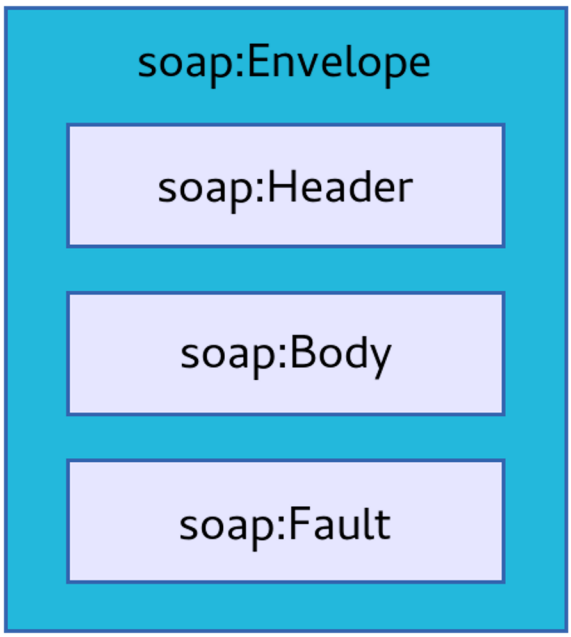
Služby běžící na těchto protokolech se liší od těch využívajících RPC (Remote Procedure Call) protokol, jenž je založen na vzdáleném volání procedur. Komunikace mezi účastníky je datacentrická – probíhá výměnou strukturovaných dat. Tyto protokoly jsou pro přenos zpráv závislé na jiném aplikačním protokolu, obvykle HTTP popř. SMTP. V případě použití HTTP protokolu nastavují minimálně hlavičku Content-Type a Content-Length.

SOAP (Simple Object Access Protocol) používá pro serializaci dat formát XML. Zpráva se skládá ze čtyř částí[6] (elementů):

[12] https://code.google.com/p/esmska/

- envelope - kořenový element pro identifikaci XML záznamu a jeho rozpoznání jako SOAP zprávy
- header - nepovinná; umožňuje libovolně rozšiřovat protokol, obsahuje tedy aplikačně závislé informace např. identifikace sezení nebo uživatele
- body - obsahuje vlastní strukturovaná data
- fault - nepovinná; popisuje v odpovědi vyskytující chybu pomocí chybového kódu, textového popisu, zdroje chyby a dalších aplikačně specifických detailů, vztahujících se k tělu SOAP odpovědi

Při serializaci dat je upřednostňováno používání elementů místo atributů, ty obsahují pouze metadata.




Obrázek 2.1: Hierarchie XML elementů uvnitř SOAP zprávy

REST (Representational State Transfer) lze považovat za jednodušší řešení SOAP varianty. Využívá převážně protokol HTTP, nikoliv jako pouhý prostředek pro přenos, jako např. v případě SOAP, ale i jako aktivní součást komunikace. Požadavek klienta se skládá ze 2-3 částí[4]:

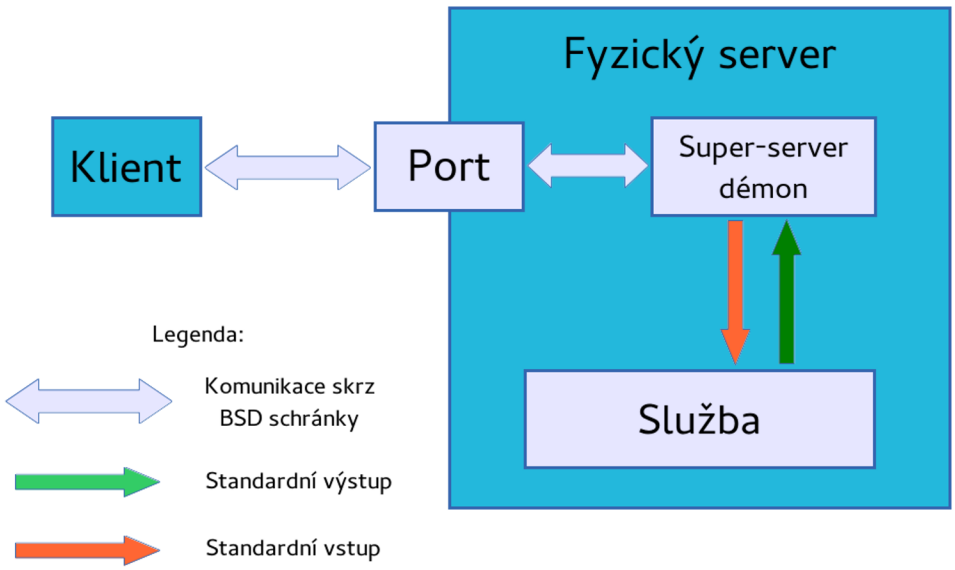
- dotazovací metoda v HTTP hlavičce – určuje požadovanou operaci, která mívá typicky shodný, ale i aplikačně specifický význam (GET, POST, DELETE, PUT atd.)
- URI zdroje v HTTP hlavičce – identifikuje objekt, na který se daná operace vztahuje (např. modul služby, záznam v databázi)
- tělo HTTP požadavku – obsahuje data v případě složitějšího požadavku, převážně v JSON formátu

Výsledkem provázání HTTP a REST protokolů jsou obecně menší zprávy, kde u jednoduchých požadavků stačí zaslat pouze HTTP metodu a URI zdroje, což je záležitost pár

desítek bajtů, k čemuž nahrazení XML formátu dat v těle požadavku či odpovědi za JSON také značně přispěje.

## 2.7 Super-server démon

Program poskytující služby, ať již přístupné z internetu nebo pouze lokálně, se nazývá démon. Typicky provádí tři typy úkonů: naslouchání na portu, přijmutí spojení a vytvoření procesu, kde původní proces pokračuje v naslouchání, a obsluha vlastního požadavku v nově vytvořeném procesu. Internetový super-server démon vykonává první a druhý typ operací, a pro obsluhu požadavku vytvoří instanci příslušného démona služby, kterému požadavek pošle. Od okamžiku spuštění démona funguje super-server jako prostředník mezi poskytovatelem služby a odesílatelem požadavku[2]. Odpadá tedy nutnost opakovaně pro každého démona stejnou síťovou část implementovat. Jelikož je služba spouštěna tzv. on-demand, nemusí být neustále puštěna v pozadí, čímž se ušetří prostor v operační paměti. Největší uplatnění super-server nachází ve vestavěných zařízeních a v zařízeních s omezenými prostředky. Hlavními zástupci jsou systemd[13], častý v desktopových linuxových distribucích, a inetd[14].




Obrázek 2.2: Schéma komunikace služby s klientem

## 2.8 Volba programovacího jazyka

Jelikož cílové zařízení je především síťový prvek, je nutné zvolit vhodný programovací jazyk – moderní, kompilovaný a rychlý. Modernost by měla zajišťovat rozsáhlou podporu různých platforem, kompilovaný charakter odstraňuje režii spojenou s převodem zdrojového kódu do vnitřní reprezentace jazyka a teprve jeho následné vykonání, rychlost je dosažena efektivitou jazyka a přítomností jen nutně potřebných abstrakcí.

[13] http://www.freedesktop.org/wiki/Software/systemd/
[14] https://www.freebsd.org/doc/en/books/handbook/network-inetd.html

Procedurální jazyk C patří mezi nejrychlejší a nejúspornější co do potřebných zdrojů. C++ je multiparadigmatický jazyk, výkonnostně srovnatelný s C, nabízející objektově orientovaný přístup. Poskytuje velice užitečné abstrakce, které usnadňují práci s dynamicky alokovanou pamětí, a generické třídy v STL (Standard Template Library) při minimální ztrátu výkonu programu. Jazyk Python, napsaný v jazyce C, patří mezi rychlejší a nejrozšířenější skriptovací jazyky. Nabízí škálu modulů s často potřebnými a užívanými funkcemi (např. pro práci se sítí), tudíž není potřeba používat externí knihovny, což přirozeně vede k méně závislostem.

## 2.9 Interpret jazyka JavaScript

Moduly programu esmska jsou napsány ve skriptovacím jazyce JavaScript. Je tedy nutno zvolit vhodný interpret tohoto jazyka a zakomponovat ho do výsledného programu. Velikánem v této oblasti je engine V8[15] napsaný v jazyce C++. Zdrojový kód skriptu je před spuštěním zkompilován do nativního strojového kódu a za jeho běhu dynamicky optimalizován. SpiderMonkey[16] patří také mezi objektové zástupce, jenž aplikuje tzv. JIT kompilaci do bajtkódu. Oba tyto enginy jsou však navrženy pro desktopové systémy, vyžadují tedy jistou míru výkonu a kapacitu operační paměti, která by mohla přesahovat prostředky dostupné na cílovém zařízení. Méně náročná implementace v C++ je TinyJS[17], která je konstruována pro jednoduchou interpretaci. Odnož tohoto projektu, 42TinyJS, obsahuje i zpracovávání regulárních výrazů, které je pro korektní práci modulů nezbytné. Pro jazyk Python se JavaScript engine jako takový mezi jeho moduly nenachází. Existují však tzv. bindings, které umožňují použít knihovny napsané v jiných programovacích jazycích. Binding je vytvořen pro engine SpiderMonkey[18], ten je však bohužel již několik let neudržovaný, a pro engine V8 zvaný PyV8[19], jehož nevýhodou je velmi strohá dokumentace, která by zkomplikovala jeho integraci, a robustnost, kterou s sebou původní engine nese. Vestavitelný interpret Duktape[20], napsaný v jazyce C, si za cíl klade co nejvyšší míru kompaktnosti – k jeho spuštění stačí necelých 50 kB operační paměti, aniž by byl funkčně ochuzen. Jeví se tedy jako nejvhodnější kandidát na aplikaci v prostředí s omezenými prostředky. Rozsáhlá a detailní dokumentace umožňuje začlenění enginu značně urychlit. Další existující řešení představuje interpret V7[21], který se snaží být také co nejúspornější, avšak v době analýzy ještě nebyl zcela hotov.

## 2.10 Síťová komunikace

Esmska moduly využívají sadu proměnných a metod pro zprostředkování komunikace a předávání dat s hlavním řídícím programem a pro poskytnutí funkcí, které samotný JavaScript nenabízí. Jedná se o metody GET a POST protokolu HTTP, prostřednictvím kterých modul s branami komunikuje. Některé brány požadují přihlašovací údaje ke zpřístupnění funkce odeslání SMS zpráv. Manipulace s těmito diskrétními daty vyžaduje úroveň zabezpečení,

[15] https://developers.google.com/v8/
[16] https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey
[17] https://code.google.com/p/tiny-js
[18] https://pydermonkey.googlecode.coom/hg/docs/rendered/index.html
[19] https://code.google.com/p/pyv8/
[20] http://duktape.org
[21] https://github.com/cesanta/v7

v tomto případě pomocí protokolu HTTPS, tedy HTTP obohaceno o šifrovací vrstvu SSL nebo TLS.

Knihovna libcurl[22] je používaná mnoha programy, ať již v desktopových nebo vestavěných zařízeních, která podporuje širokou škálu protokolů, do nichž patří i protokol HTTP a HTTPS. Šifrovací část HTTPS je realizována buď pomoci OpenSSL[23] nebo mbed TLS[24] (dříve pod názvem PolarSSL). Libcurl je napsaná v jazyce C, bylo by ji však možné použít i v případě zvolení jazyka C++. Alternativní řešení pro C++ představuje extenzivní sada knihoven Boost[25], která obsahuje část věnující se síťovým přenosům a obalení komunikačních schránek SSL vrstvou. Zde však není u realizace šifrování možnost volby – je potřeba použít OpenSSL implementaci. Python nabízí v této problematice nejpříznivější řešení. Prostředky ke komunikaci protokolem HTTP se nachází přímo v modulu jazyka. Je-li interpret Pythonu přeložen s podporou SSL, je možné nachystané třídy a jejich metody použít k uskutečnění spojení HTTPS protokolem.

[22] http://curl.haxx.se/libcurl/
[23] https://www.openssl.org
[24] https://tls.mbed.org
[25] http://www.boost.org

# Kapitola 3

# Návrh

## 3.1 Zvolené prostředky

Cílové zařízení je primárně směrovač s operačním systémem OpenWrt. Při těchto podmínkách se od programu očekává, že bude napsán úsporně, kvůli omezení dostupných hardwarových prostředků, a přenositelně, neboť architektury různých síťových prvků se mohou značně lišit. Aspekt přenositelnosti nejvíce přísluší skriptovacímu jazyku Python. Práce se síťovými přenosy by byla snadná, avšak nedostupnost nativního modulu pro interpretaci JavaScriptu a tedy nutnost použít některý s bindingů, který není určen pro tento typ aplikací, činí tento jazyk pro řešení jako nevhodný. Přidáme-li fakt, že skriptovací jazyk lze těžko pro jakoukoliv doménu optimalizovat, můžeme Python z možných jazyků zcela vyloučit. Systém OpenWrt poskytuje pro překlad a tvorbu vlastních balíčků mimo oficiální repozitář tzv. toolchain, tedy sbírku utilit a nástrojů, která obsahuje především pro nás důležitý překladač nejen jazyka C a C++. Existuje-li tedy OpenWrt pro danou architekturu, je také zajištěna přítomnost překladače pro tuto architekturu. Uvážíme-li použití jazyka C++, zacházení s dynamicky alokovanou pamětí je přívětivé díky kontejnerům a chytrým ukazatelům. JavaScriptové moduly využívají regulárních výrazů nad obsahem webové stránky např. ke kontrole přihlášení uživatele. Je tedy nutné použít interpret 42TinyJS, který implementaci regulárních výrazů obsahuje. Po vytvoření jednoduchého programu na účely otestování a seznámení se s knihovnou vyvstaly nečekané problémy. Knihovna interně pracuje s ukazateli na datový typ `void`, které jsou přetypovávány na `int`, tedy na datový typ o minimální délce 32 bitů, neboli 4 bajty. Kdyby proběhlo přetypování na stroji s 64 bitovou architekturou, došlo by k ořezání horních 32 bitů, což by mohlo způsobit ztrátu informace. V blízké budoucnosti, kdy i síťové prvky budou stavěné na 64 bitové architektuře, by ořez zapříčinil problémy. Náprava vyžaduje zásah do zdrojového kódu, který by bylo nutné provést při použití každé nové verze interpretu k překladu programu, na několika místech knihovny. Další řešení je informování o této skutečnosti autora 42TinyJS a požádání o aplikaci změny. Zásadnější problém představuje velikost spustitelného souboru testovacího programu, která činí přibližně 1,9 MB. Pro cílové zařízení je takový program nevhodný a příliš omezující. Po zapnutí optimalizací překladače pro velikost výsledného souboru se program zredukoval na přípustných 480 kB. Předpoklad pro nejmenší velikost souboru má implementace v jazyce C za použití interpretu Duktape. Funkčně ekvivalentní testovací program je téměř o polovinu menší, zabírá okolo 260 kB. Jazyk C a interpret Duktape se tedy jeví jako nejvhodnější řešení – nejlépe splňují požadavek na nejmenší velikosti spustitelného souboru. Použitím tohoto jazyka bude také možné využít standardní knihovnu uClibc.

## 3.2 Struktura

Konečný program bude čistě konzolový. K jeho ovládání bude dostupné tzv. command-line interface (zk. CLI), tedy příkazová řádka. Jelikož v cílovém prostředí není k dispozici žádné grafické rozhraní, není možno interaktivně zobrazovat bezpečnostní CAPTCHA obrázky proti robotům, z čehož vyplývá, že je nemožné využívat webové SMS brány, které toto zabezpečení používají. Brány, které CAPTCHA mechanismus nevyužívají, vyžadují ve většině případů registraci, což ve své podstatě anti-robotový systém nahrazuje. K odeslání SMS zprávy je nutno se k bráně přihlásit, díky čemuž je odesílatel do jisté míry identifikován, a v případě nadměrného užívání či zneužívání k rozesílání spamu nebo dalších nežádoucích zpráv lze účet blokovat. Samotné vytvoření účtu již CAPTCHA obrázek obsahuje.

Program se bude dělit na 4 hlavní části. První část představuje zpracování parametrů, které zahrnuje kontrolu jejich přítomnosti, syntaktickou a sémantickou správnost. Další část zpřístupňuje pro JavaScript moduly potřebné proměnné získané z parametrů při spuštění, pomocí kterých je daný modul parametrizován, a funkce, které slouží ke komunikaci s hlavním řídícím programem a k vykonávání operací, jenž nejsou součástí samotného JavaScript jazyka. Mezi nejdůležitější funkce patří HTTP(S) operace GET a POST, které budou implementovány pomocí knihovny libcurl. Třetí část je prakticky zcela mimo režii hlavního programu. Jedná se o samotnou interpretaci JavaScript modulů, která probíhá zcela autonomně. Poslední část analyzuje výsledek interpretace modulu, vyhodnocuje případné chyby, o nichž informuje uživatele, a provádí úklid před ukončením programu.

Vedle hlavního programu bude dostupná i webová služba, umožňující odesílat SMS zprávy vzdáleně. Přístup k službě bude zajišťovat REST rozhraní. SOAP, používající XML k serializaci strukturovaných dat, je zbytečně robustný. K zaslání jednoduchého požadavku je potřeba obalit zprávu mnoha metadaty. Výřečnost XML také odpovídá velikosti potřebného parseru pro načtení dat ze zprávy a velikosti části věnující se vytvoření odpovědi z celkového zdrojového kódu služby. Zvolení REST rozhraní je pro tuto aplikaci vhodnější, díky své přímočarosti, lepšímu využití prvků HTTP protokolu a především kratším zprávám ve formátu JSON, čímž se redukuje režie zpracování požadavků a vytváření odpovědí. Kompaktnější forma komunikace také přispívá k menším datovým tokům. Služba je ve své podstatě pouze zpřístupnění funkcí programu z vnějšího prostředí, stará se tedy jen o síťovou komunikaci, dekódování informací zaslaných klientem a jejich nachystání do požadovaného formátu pro výkonnou část, po jejímž dokončení proběhne opět vyhodnocení chyb a průběhu interpretace JavaScript modulu. Vyhodnocení je posléze zformátováno a sestaveno do odpovědí, které jsou záhy klientovi odesláno. Jelikož samotný proces odesílání SMS zpráv je identický s tím v hlavním programu, může být sdílen. Z tohoto důvodu je velice praktické psát hlavní program modulárně. Zasílání SMS zpráv bude tvořit kolekci hierarchicky uspořádaných funkcí, které bude možno exportovat do knihovny, kterou bude jak hlavní program, tak i webová služba sdílet a volat její funkce. Tento přístup o polovinu zredukuje celkový objem kódu v případě využití obou programů. Hlavním účelem programu je zpřístupnění funkce posílání SMS zpráv z příkazové řádky a zevnitř skriptů při zachování nízkých hardwarových nároků. Webová služba by umožňovala odchozí body centralizovat a tím odstranit v jiných případech nezbytnou distribuci programu v rámci sítě, čehož lze uplatnit i při monitorování sítě k zasílání urgentních zpráv, které vyžadující okamžitou pozornost, jejímu správci. Jelikož požadavek pro službu obsahuje diskrétní data jako přihlašovací údaje, v normálních situacích je potřeba takovou komunikaci zabezpečit. Rozhraní služby však bude přístupné pouze v rámci lokální sítě, nikoliv skrz internet. Není tedy potřeba se zabezpečením komunikace klienta a služby zabývat.

# Kapitola 4

# Implementace

Jádro procesu odesílání SMS zpráv tvoří jednotlivé JavaScript moduly, řídící program pouze vytváří prostředí k jejich spuštění a poskytuje jim dodatečné prostředky. Náročnost a rychlost běhu celého programu tedy značně závisí na formě zápisu daných modulů. Při implementaci řídící části bude nejpřínosnější se soustředit na redukci paměťové zátěže, neboť v průběhu činnosti programu neprobíhají žádné natolik složité výpočty, při kterých by jejich optimalizace vykazovala znatelný rozdíl.

## 4.1 Zpracování parametrů

Vstupní parametry jsou získávány ze standardního vstupu při spuštění programu. Mezi ty základní zajisté patří telefonní číslo mobilního zařízení příjemce SMS zprávy a samotný text zprávy. Dále je vyžadován název JavaScript modulu, který obsluhuje danou webovou bránu. Moduly jsou hledány ve standardním Unixovém adresáři s cestou `/usr/share/esmska`. V případě potřeby má uživatel možnost určit adresář s uloženými moduly, ve kterém bude program předpokládat, že se zvolený modul nachází. Program přijímá dva nepovinné, avšak ve většině případů vyžadované, parametry – přihlašovací jméno a heslo. Pokud nebudou zadány, budou přihlašovací údaje reprezentovány prázdnými řetězci, což v případě, že jsou údaje pro korektní fungování modulu nezbytné, bude mít posléze za následek ukončení programu s chybou informující o nezdařeném přihlášení. Jedna webová brána, u které se není potřeba k jejímu používání registrovat, poskytuje možnost uvedení telefonního čísla odesílatele zprávy. K dispozici je také krátká nápověda vysvětlující uživateli význam jednotlivých parametrů.

Po spuštění programu podstupují parametry první fázi kontroly, tedy analýzu jejich přítomnosti a syntaktické správnosti. Mohou být zadávány v libovolném pořadí. Každý parametr je vždy jednoznačně identifikován přepínačem, za kterým následuje argument. Pokud se budou nacházet dva přepínače za sebou, bude druhý přepínač považován za argument prvního. Chybí-li argument k některému přepínači, tedy je-li celkový počet zadaných parametrů lichý, je uživatel o této skutečnosti bezprostředně upozorněn. Kontrola přítomnosti parametrů je omezena pouze na ty, které jsou nezbytné pro správné fungování všech modulů. Jedná se o telefonní číslo příjemce, vlastní text SMS zprávy a název JavaScript modulu. Situace, kdy by název modulu chyběl, by byla podchycena v pozdějších stádiích běhu programu. Zbytečně by se však inicializoval interpret JavaScriptu, který by byl záhy opět ukončen. Takto je chyba eliminovaná bez zbytečných prodlev. U telefonního čísla a textu zprávy by se reakce v případě vynechání této kontroly odvíjela od implementovaných

kontrolních schopností samotné webové brány. Nezadané položky by opět byly nahrazeny za prázdné řetězce. S vysokou pravděpodobností mají všechny brány ošetřenu situaci prázdného pole s telefonním číslem příjemce. Otázkou však je, jestli brány akceptují prázdnou zprávu jako validní, nebo zda ji odmítnou s upozorněním na danou skutečnost. Další zdroj pochybností, je-li ošetření obou případů přítomno, představuje umístění této kontroly. Pokud je aplikována pouze u textových polí formuláře na webovém rozhraní brány, při odesílání zprávy se modul kontrole kompletně vyhne, neboť samotný formulář nevyužívá, pouze data odešle HTTP metodou POST na odpovídající webovou adresu. Je tedy žádoucí provést opatření, jenž tyto problémy a případné nedefinované chování vylučují.

V druhé fázi zpracovávání probíhá ověření sémantické správnosti, i když jej nelze považovat jako zcela samostatnou část. Probíhá totiž postupně během vykonávání programu. Jako první je prověřováno, zda se modul se zadaným názvem nachází ve výchozím umístění či v umístění zadaném uživatelem. Posléze dojde ke zkontrolování přihlašovacích údajů, jsou-li požadovány ze strany webové brány. Poslední kontrolní bod typicky představuje ověření, zda zadané telefonní číslo příjemce spadá do mobilní sítě, do které webová brána umožňuje zasílat SMS zprávy. Každý mobilní operátor disponuje sadou předvoleb, jenž tvoří první tři číslice, které telefonním číslům svých klientů přiřazuje. Podle těchto předvoleb lze tedy snadno všechny přední mobilní poskytovatele rozpoznat.

## 4.2 Minimální kostra pro spuštění JavaScriptu modulů

JavaScript engine Duktape funguje na principu tzv. kontextů. Při inicializaci interpretu je vytvořen kontext, jenž reprezentuje programový zásobník. Počet vytvořitelných kontextů není omezen, v tomto případě je však jeden plně dostačující. Po ukončení využívání kontextů je zapotřebí veškeré vytvořené kontexty zdestruovat pomocí příslušné funkce, aby nedocházelo k případným únikům paměti. Program bude spouštěn jednorázově, nepoběží tedy dlouhodobě, z čehož lze nebezpečí úniku paměti vyloučit a spoléhat na uvolnění správou paměti operačního systému. Jedná se spíše o korektní postup při zacházení s dynamickou pamětí a prostředky obecně – po skončení jejich využívání zajistit navrácení.

Ze vstupních parametrů je získán název JavaScript modulu, ke kterému je přidána přípona `.gateway`, jenž všechny moduly obsahují, a celý tento název je konkatenován s výchozím nebo uživatelem zadaným umístěním modulů. Tímto je vytvořena cesta k modulu, ať již absolutní nebo relativní, kterou je možno použít jako parametr pro jednu z Duktape funkcí. Ta je schopna v rámci konkrétního kontextu vyhodnotit obsah předaného souboru jako kus zdrojového kódu jazyka JavaScript. Návratová hodnota funkce indikuje úspěch či neúspěch interpretace souboru. V případě úspěchu je na kontextu, neboli na zásobníku, ponechán výsledek interpretace. Jelikož moduly obsahují pouze sadu definic funkcí, žádnou smysluplnou nebo významnou hodnotu nelze očekávat. Na zásobníku je ponechána hodnota typu `undefined`, což lze v tomto případě v jazyce C připodobnit k datovému typu `void`. Soubor je považován za jednu velkou funkci, která může a nemusí mít návratovou hodnotu, stejně jako funkce, popř. procedury, v imperativních a jiných programovacích jazycích. V případě neúspěchu interpretace je na zásobníku uložen objekt typu `Error`, ze kterého lze pomocí jeho atributů získat informace o problému, jenž se během vyhodnocování souboru vyskytl.

Pro manipulaci s entitami v Duktape kontextu je potřeba znát jejich aktuální umístění neboli index v rámci kontextu. Entity, které jsou použity jako parametry funkcí, jsou lokalizovány právě pomocí indexu v zásobníku. Některé funkce vyžadují pozici absolutní, kde první entita se nachází na nultém indexu, jiné relativní od vrcholu zásobníku, čiže indexy, jsou záporné. Mnoho jich také operuje stejně jako instrukce jazyka symbolických instrukcí,

dnes již zastaralých zásobníkových architektur, nebo instrukce pro práci s Floating Point Unit (zk. FPU), tedy s výpočetní jednotkou pro čísla s plovoucí řádovou čárkou. Parametry funkcí jsou totiž implicitní, stejně jako operandy u zmiňovaných instrukcí. Po inicializaci interpretu je kontext prázdný. Po evaluaci JavaScript modulu, je na vrcholu zásobníku, momentálně s indexem nula, zanechána návratová hodnota modulu. Jelikož je v případě úspěchu nepotřebná, můžeme ji v zásobníkových strukturách známou operací `pop` odstranit. Každý modul obsahuje bezparametrickou funkci `send`, která danou webovou bránu obsluhuje, a kterou je nutno z kontextu zavolat. Všechny funkce přítomné v modulu jsou v tomto bodě již vyhodnoceny a jsou přístupné z globální úrovně vnořeného programu. Jelikož všechny entity jsou v JavaScriptu objekt, výjimkou není ani ona globální úroveň, která je reprezentována tzv. superglobálním objektem. Duktape poskytuje funkci, která tento objekt vloží na vrchol kontextu, čímž všechny vyhodnocené funkce a proměnné zpřístupní. Všechny entity, neboli subobjekty, jsou součástí superglobálního objektu – jsou jeho atributy. Zde přichází na řadu funkce manipulující s obsahem zásobníku. Pro získání hodnoty určitého atributu je nejdříve potřeba na zásobník vložit jméno atributu v podobě textového řetězce. Poté je zavolána funkce, která implicitně interpretuje entitu na vrcholu zásobníku jako název atributu, a jako parametr přijímá relativní index objektu, kterému požadovaný atribut náleží. Je-li specifikovaný atribut v objektu přítomen, je jeho hodnota vložena na vrchol zásobníku na místo řetězce se jménem. Stejný postup provází i zpřístupnění funkce, neboť se jedná o atribut typu `function`, nad kterým je aplikován operátor volání funkce kulaté závorky, jako v mnoha jiných jazycích. Jakmile je funkční objekt na zásobníku, stačí zavolat příslušnou Duktape funkci, která se opět implicitně pokusí objekt na vrcholu zavolat jako funkci. Při volání lze také specifikovat počet parametrů, které jsou na zásobníku pro funkci dostupné. Parametry je nezbytné umístit ještě před samotnou funkci. Zde však žádné parametry není nutno poskytovat. Vykonávání JavaScript funkcí je v plné režii interpretu. Samotná funkce volající `send` z modulu opět návratovou hodnotou informuje o zdařilosti průběhu funkce. Po vrácení řízení hlavnímu programu je na zásobníku místo funkčního objektu ponechána návratová hodnota funkce nebo chybový objekt, dle úspěšnosti interpretace.

## 4.3 Rozšíření kostry o vyhodnocení výsledku a chyb

Po vytvoření základního prostředí pro běh JavaScript modulů je potřeba jej vylepšit o zpracování zanechané hodnoty nebo chybového objektu na vrcholu Duktape kontextu. Modulová funkce `send` vrací hodnotu typu `boolean` – `true` v případě úspěšného odeslání SMS zprávy, `false` kdyby nastaly potíže a odeslání se nezdařilo. Nepodaří-li se zprávu odeslat, pravděpodobně nastal problém spojený s webovou bránou, ať již špatné přihlašovací údaje, telefonní číslo příjemce nebo nefunkční samotná webová brána. V každém případě je dostupný minimálně chybový kód, a v některých případech i doplňující hláška s podrobnějšími informacemi, díky kterým lze alespoň na základní úrovni problém identifikovat. Je doporučeno nespoléhat na dodržení stanovených směrnic autory modulů a programovat tzv. defenzivně, tedy počítat s možností, že k chybě může dojít i v místech se smluvenými konvencemi. Jedním takovým místem je datový typ návratové hodnoty. Funkce `send` může vracet číslo, které správně indikuje stav odesílání zprávy, ale až po jeho přetypování. Konkrétně nula odpovídá po přetypování hodnotě `false` a číslo větší popř. i menší než nula je interpretováno jako hodnota `true`. Je vhodné o této operaci uživatele informovat, neboť nastane-li případ, kdy `send` vrátí hodnotu jiného datového typu než číselného nebo `boolean`, může dojít ke konverzi, kdy přetypovaná hodnota neodpovídá skutečnému stavu

odesílacího procesu, což vede k dezinformaci uživatele. Je-li si uživatel po přečtení varování o konverzi nejistý nebo je-li přesvědčen o chybném ohlášení stavu, může konkrétní úsek v modulu zkontrolovat a ujistit se o jeho správnosti nebo případnou nesrovnalost napravit.

Druhý případ představuje situace, kdy selže interpretace funkce `send` nebo při počáteční evaluaci celého modulu. Při naskytnutí některého z těchto problémů vrátí vyhodnocovací funkce modulu nebo funkce spouštěcí modulovou funkci `send` nenulovou hodnotu a na vrcholu zásobníku ponechá odpovídající chybový objekt. Nejpravděpodobnější chyby souvisí se syntaktickou stránkou modulu. Ty jsou odhaleny při prvotním vyhodnocení modulu. Sémantické chyby, týkající se především používání nedefinovaných proměnných a funkcí, se projeví až při zavolání dotyčné funkce, která chyby obsahuje. Všechny vyvstalé problémy jsou uživateli samozřejmě reportovány. To však k jejich rychlému opravení bohužel nestačí. Jazyk JavaScript, co se detailních chyb týče, není příliš sdílný a ve složitějších konstrukcích nijak nepomáhá zdroj selhání identifikovat.

Chybový objekt je při interní chybě JavaScriptu tzv. vyhozen. Jedná se o mechanismus výjimek. Objekt prochází skrz všechna zanoření, ve kterých se program momentálně nachází. Není-li v průběhu objekt patřičnou konstrukcí jazyka zachycen a dostane se na globální úroveň, je program ukončen a v tomto případě je objekt uložen na vrcholu zásobníku. Chybové objekty je možné programově generovat, což umožňuje informovat o výskytu chyb jiných než jen interpretačních. Mechanismus výjimek bude klíčový k oznamování a rozeznávání chyb, jenž mohou nastat v rozšiřujících funkcích užívaných JavaScript modulem. Tato část, starající se o chyby při interpretaci, bude tedy v pozdějším stádiu vývoje rozšířena.

## 4.4 Implementace rozšiřujících funkcí užívaných uvnitř JavaScript modulů

Interpret Duktape poskytuje možnost volání nativních funkcí jazyka C zevnitř interpretovaného JavaScript programu. Díky této schopnosti lze implementovat potřebné rozšiřující funkce. Všechny tyto nativní funkce mají jednotně stanovený formát prototypu, který musí dodržet, aby je bylo možné z modulu volat. Návratová hodnota funkcí je číselného datového typu `duk_ret_t`, pod kterým se skrývá `int`. Parametr funkce přijímají pouze jeden, a to ukazatel na Duktape kontext. Každá úroveň zanoření při volání funkcí vytváří vlastní kontext, tedy po předání řízení C funkci je pracovní zásobník zcela prázdný. Požadujeme-li přistoupit ke globální úrovni JavaScript programu, je možno si superglobální objekt vyžádat a uložit na vrchol zásobníku. Situace je však jiná, pokud je funkce v Duktape interpretu registrována jako funkce s parametry. V tom případě jsou předané parametry funkci dostupné na jejím pracovním zásobníku přesně v pořadí, v jakém byly uspořádány při zavolání funkce. Je-li funkci předán větší počet parametrů než je u funkce registrováno, jsou přebytečné parametry oříznuty a zahozeny. V případě opačném, kdy je funkci předán nedostatek parametrů, je zbytek chybějících doplněn hodnotami `undefined`. Tyto funkce mohou samozřejmě také vracet hodnoty, k čemuž se opět využívá zásobník. Stačí na vrchol vložit hodnotu, jež má být výsledkem funkce, a jako návratovou hodnotu uvnitř nativní funkce zvolit číslo, které odpovídá počtu návratových hodnot připravených na zásobníku. Je tedy možné vracet více než jen jednu hodnotu. Je-li návratové číslo záporné, je vyhozena výjimka odpovídající danému číslu. Jedná se o zkrácení procesu generování předdefinovaných chybových objektů.

Soubor rozšiřujících funkcí se dělí na dvě skupiny. První zahrnuje ty, jenž zprostředkovávají komunikační kanál směrem z modulu do hlavního řídícího programu. Druhá skupina

obsahuje ty, které poskytují modulům schopnosti přesahující specifikaci JavaScriptu podle ECMAScript[1] normy. Všechny funkce jsou z hlediska modulu metody globálního objektu `EXEC`.

### 4.4.1 Funkce `setProblem` a použití výjimek

První komunikační funkce se nazývá `setProblem`. Dojde-li při procesu odesílání SMS zprávy, konkrétně při komunikaci s webovou bránou, k chybě, vrátí funkce `send` hodnotu false a informuje hlavní program. Účelem funkce `setProblem` je právě nastavení příslušné proměnné na hodnotu chybového kódu odpovídající nastalé chybě a případné doplnění o text podrobněji komentující daný problém. Funkce přijímá jeden nebo dva parametry. První reprezentuje chybový kód problému. Tyto chybové kódy jsou seskupeny do výčtového typu, ve kterém jsou všechny pojmenovány. Parametr lze tedy očekávat buď v podobě číselné konstanty nebo textového řetězce obsahující název chyby. V případě obdržení řetězcové formy je třeba parametr porovnat se všemi názvy obsaženými ve výčtovém typu. Porovnáván je vždy minimální počet znaků potřebný k jednoznačnému rozlišení jednotlivých chybových stavů. Po aplikaci defenzivního přístupu byla přidána varianta pro situaci, kdy není parametr identifikován jako validní chybový kód z výčtového typu – chybě je přiřazena konstanta odpovídající neznámé chybě. Druhý parametr obsahuje textový řetězec detailněji popisující příčinu selhání odeslání zprávy. Zde je zřejmá jedna z výhod objektově orientovaného jazyka. Původní implementace v Javě využívá mechanismus přetěžování funkcí, díky kterému je možno definovat více funkcí se stejným jménem. Liší se pouze v počtu nebo datových typech parametrů. Jelikož je doplňující text dostupný jen při určitých typech chyb, je metoda `setProblem` volána buď s jedním nebo se dvěma parametry, k čemuž se přetěžování funkcí přímo nabízí. Jazyk C objektově orientovaný není, tudíž tento způsob řešení není možné aplikovat. Funkce `setProblem` bude pouze jedna a bude přijímat dva parametry. V případě, kdy bude funkce zavolána pouze s prvním parametrem, využijeme vlastnosti Duktape enginu vysvětlenou dříve – druhý parametr bude doplněn hodnotou `undefined`. Je vhodné kontrolovat rozpoznaný chybový kód, neboť dojde-li k situaci, ve které chybový kód implikuje přítomnost dodatečného textového řetězce, ale druhý parametr obsahuje pouze hodnotu `undefined`, je nezbytné o této skutečnosti uživatele informovat a případně interpretaci modulu ukončit. Další překážku představuje pevně stanovený prototyp funkcí – nelze předávat vlastní parametry dovnitř funkce. Z toho vyplývá, že všechny datové struktury ze strany hlavního řídícího programu, se kterými je potřeba uvnitř funkcí pracovat, musí být přístupné z globální úrovně. K uložení chybového stavu a textu je k dispozici struktura, obsahující proměnnou zmiňovaného výčtového datového typu a ukazatel, který odkazuje na případný blok paměti s uloženou detailní hláškou. U předávaného parametru není v tomto případě potřeba kontrolovat, zda se jedná opravdu o textový řetězec, neboť je na něj aplikována vestavěná funkce JavaScriptu `toString`, která jakýkoliv objekt či hodnotu převede do odpovídající textové podoby. Není tedy možné, aby došlo k havárii programu nebo nedefinovanému chování. V nejhorším případě bude na výstupu nesmyslná chybová hláška, díky které bude možné chybnou část modulu dohledat.

Ve funkci `setProblem` může dojít k chybové situaci, kterou je třeba hlavním řídícím programem vyhodnotit, což vede k rozšíření části kostry programu zpracovávající chybové stavy, jak již bylo zmíněno v sekci 4.3. Bude opět využito mechanismu výjimek, neboť výjimka nemusí být nutně datového typu `Error`. Tzv. vyhodit lze hodnotu libovolného datového typu, díky čemuž je možné chyby interpretace a chyby pocházející z rozšiřují-

[1] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=55755

cích funkcí snadno rozlišit. Vzhledem k počtu možných interních chyb postačuje ke sdělení typu chyby jediná číselná hodnota. Knihovna zprostředkovávající síťovou komunikaci libcurl používá také číselné hodnoty k identifikování chyb, které je posléze možné převést do textové podoby. Tyto chybové kódy bude také možno posílat ve výjimkách ke zpracování. K rozlišení interních chyb a chyb knihovny libcurl budou hodnoty interních chyb začínat číslem o jedna větším než největší hodnota libcurl chyby.

### 4.4.2 Funkce `setSupplementalMessage`

Druhá a poslední komunikační funkce zprostředkovávající jednosměrný komunikační kanál z modulu se nazývá `setSupplementalMessage`. Slouží k předání dodatečné zprávy od webové brány skrz JavaScript modul do hlavního řídícího programu, kde je posléze zobrazena uživateli. Zprávy předané touto funkcí jsou informativního charakteru. Obsahem může být zmiňovaná doručenka, nabízí-li webová brána tuto funkci, nebo v případě placených bran cenu odeslané SMS zprávy a kreditový zůstatek na účtu. Opět zde není nutné parametr kontrolovat na datový typ, neboť s využitím vestavěné funkce `toString` je výsledkem textový řetězec vždy. V případě konzolového programu by nejen tento typ zpráv směřovaný uživateli postačovalo vypsat na standardní nebo standardní chybový výstup. Jelikož však bude jádro programu využívat i služba, která tyto hlášky bude odesílat vzdálenému uživateli či skriptu, je zapotřebí všechny tyto zprávy uchovávat, aby byly po skončení interpretace modulu nadále dostupné. K jejich uložení je nutné vytvořit patřičnou datovou strukturu, opět na globální úrovni hlavního řídícího programu kvůli prototypu rozšiřujících funkcí. Datovou strukturu by bylo vhodné navrhnout tak, aby byla schopna pojmout jak hlášky informativní povahy, tak i ty chybové a varovné. Informativní zprávy bývají často jednolité. U chybových je situace odlišná, neboť typická zpráva tohoto typu sděluje uživateli typ chyby, místo nebo oblast, kde došlo k chybě, která je následována vlastní sdělnou zprávou. Datová struktura tedy musí být schopna uchovat jednotlivé částí zpráv, které je možno jednoduše spojit do kompletní zprávy, a je nutné označit vždy poslední části k odlišení dvou sousedících zpráv. Výsledná struktura se skládá z pole struktur, v němž se nachází jednotlivé části zpráv, a z číselné proměnné udávající index první volné struktury v poli k uložení části zprávy. Vnitřní struktura v poli obsahuje ukazatel na paměťový blok s daným kusem zprávy, proměnnou výčtového typu k rozeznání typu úryvku a proměnnou typu `boolean` k určení, zda je konkrétní část v rámci zprávy poslední. Výčtový typ by bylo možné nahradit typem `boolean`, neboť nabývá pouze dvou hodnot. Sdružení spolu logicky souvisejících hodnot do výčtového typu a jejich pojmenování však umožní čtenáři zdrojového kódu snadněji pochopit jejich význam. Alternativní řešení neposkytuje žádné benefity ani z hlediska paměťové úspory. Vnitřní struktura obsahuje ukazatel, který, bereme-li v úvahu 32 bitovou architekturu procesoru, v paměti zabírá 4 bajty. Jelikož jsou ve struktuře přítomny další položky, celková velikost struktury je rozšířena. Překladač gcc užívaný k tvorbě balíčků pro OpenWrt systémy se snaží velikosti struktur automaticky zarovnávat. V případě 32 bitových architektur na násobek čtyř v bajtech. Výsledná struktura bude mít tedy velikost 8 bajtů, které proměnnou výčtového typu a typu `boolean` bez problému obsáhnou. Pro velikost pole těchto struktur je hodnota 4 postačující, neboť v krajním případě bude v poli uložena jednolitá informativní zpráva a chybová hláška skládající se z maximálního počtu 3 částí, odkud získáme zvolenou velikost pole.

Při implementaci předchozí datové struktury bylo zřejmé, že všechny hlášky pocházející z JavaScript modulů budou muset být kopírovány do dynamicky alokovaných bloků paměti, což naznačovalo značné komplikace při rozlišování dynamických a statických chy-

bových hlášek a uvolnění těch správných. Statické chybové zprávy jsou pohromadě uloženy v poli textových řetězců opět na globální úrovni. Tímto jsou dostupné odkudkoliv a hlavně centrálně. Nedochází tedy k žádným zbytečným duplikacím. Při zkoumání vnitřních procesů Duktape enginu se však projevila velice užitečná vlastnost. Je-li na zásobníku vložen řetězec, třeba jako parametr, lze na něj získat konstantní ukazatel. Zůstane-li JavaScript kontext po skončení interpretace netknutý, jsou všechny takto získané ukazatele stále platné, není-li obsah paměti smazán nebo přepsán jinými daty. Jelikož se nachází nastavování informačních a chybových hlášek vždy ke konci interpretace, riziko přepsaní či ztráty obsahu odpadá. Díky této skutečnosti není třeba žádnou z hlášek kopírovat, stačí přiřadit příslušné ukazatele, což vede k redukci režie práce s pamětí a celkovému zjednodušení architektury starající se o textové výstupy.

### 4.4.3 Funkce `sleep`

První skutečně rozšiřující funkce se nazývá `sleep`. Prakticky je použita pouze v jediném modulu, a to na zjištění stavu odesílané SMS zprávy. Webová brána totiž umožňuje po určitou dobu zpětně zjistit, zda byla zpráva úspěšně odeslána. Proces odesílání není okamžitý a může chvíli trvat a právě pro tento případ je vytvořena funkce `sleep`. Po zaslání zprávy bráně je modul na krátkou dobu pozastaven, neboli uspán, a poté pomocí identifikačního čísla zprávy je od webové brány vyžádán stav odesílání. Je-li SMS zpráva stále v procesu odesílání, je modul opět pozastaven a posléze stav aktualizován. Tento cyklus je opakován maximálně 10x. Funkce `sleep` je tedy velice primitivní, skládá se pouze z kontroly parametru, jenž je funkci předáván, a volání jádra systému k suspendování procesu. Jazyk C nabízí u uspávání procesů 2 úrovně přesnosti. Doba odstavení procesu může být specifikována v sekundách nebo v mikrosekundách. Modulová funkce `sleep` však očekává dobu v milisekundách, je tedy potřeba použít variantu s mikrosekundami a patřičně předanou hodnotu upravit. Jelikož JavaScript nerozlišuje datové typy celých a reálných čísel, je nutné parametr explicitně přetypovat na celočíselný datový typ pro případ, že by v modulu bylo předáno reálné číslo. V případě, že parametr není číselného datového typu vůbec, je vyhozena číselná výjimka odpovídající této chybě.

### 4.4.4 Funkce `getUrl`

Mezi nejdůležitější funkce nepochybně patří `getUrl`. Jejím úkolem je stáhnout obsah webové stránky pomocí HTTP operace GET podle zadané URL adresy. Zde přichází na řadu knihovna libcurl. Než je možné knihovnu aktivně uvnitř funkce používat, je nezbytné ji inicializovat, což zahrnuje vytvoření ukazatele často označovaného jako tzv. handle, který bude reprezentovat samostatné okno webového prohlížeče zprostředkovávající komunikaci HTTP a HTTPS protokolem. Funkce bude používána v rámci interpretace modulu, z čehož vyplývá, že vytvoření handlu je potřeba provést ještě před spuštěním modulové funkce `send`. Jelikož je u libcurl funkcí potřebné identifikovat instanci přenosové knihovny, ke které se má požadovaná operace vztahovat, vyžadují funkce přítomnost handlu na první pozici v parametrech. Jak již víme, prototyp nativních rozšiřujících funkcí nedovoluje předávat vlastní parametry. Z toho plyne, že libcurl handle bude musí být opět přístupný z globální úrovně hlavního řídícího programu. Platnost handlu stačí udržet do konce interpretace, narozdíl od Duktape kontextu, u kterého se pracuje s referencemi v něm obsaženými, takže musí zůstat vytvořený i po dokončení evaluace modulové funkce `send`. Parametry funkce `getUrl` jsou celkem dva. První udává URL adresu webové stránky v textovém řetězci, jejíž obsah je požadován. Druhý parametr je nepovinný, může obsahovat prázdné pole nebo nemusí být předán

vůbec. V případě, že parametr je pole, které je neprázdné, jsou jeho prvky považovány za parametry, s nimiž má být požadavek GET odeslán. Za sebou jdoucí dvojice položek v poli představují vždy atribut a jeho hodnotu. Celkový počet prvků v poli tedy musí být nutně sudý. Není-li tomu tak, je vyslána výjimka s patřičným chybovým kódem. Parametry předávané přes webovou adresu jsou však kódovány v application/x-www-form-url-encoded, je tedy potřeba položky pole, není-li pole prázdné, do tohoto formátu dodatečně převést, o což se stará funkce `appendParameters`. Je jí předána webová adresa stránky a pole s parametry, a jako výsledek vrátí dynamicky alokovaný řetězec, který obsahuje webovou adresu spolu s korektně spojenými parametry, který je třeba po získání obsahu stránky nutno uvolnit. Při tvorbě tohoto řetězce dochází k postupnému zvětšování paměťového bloku. Pokud by nebyl dostatek volné paměti při jednom ze zvětšování, je stávající blok uvolněn a poslána výjimka. Nyní zbývá připravit prostředky pro samotné stažení obsahu webové stránky. Pokud nechceme obsah vypsat na standardní výstup, ale uložit ho do paměti, umožňuje libcurl definovat tzv. callback funkci se specifickým prototypem, a té bude předán vždy kus stažených dat k vlastnímu zpracování. Funkci je také možné předat ukazatel na libovolnou datovou strukturu, do níž lze data zapisovat. Obsah stránky se vkládá do dynamického bloku paměti, který je podle potřeby zvětšován. Může tedy nastat situace, kdy již není dostatek paměti. I zde je tento problém řešitelný s využitím JavaScript výjimek. Alokování paměti probíhá uvnitř callback funkce. Její návratová hodnota udává počet bajtů, které zpracovala. Je-li tato hodnota menší než počet bajtů předaných dat, přenos stránky je ukončen s určitým chybovým kódem, podle kterého lze tuto situaci rozpoznat a výjimkou na ni reagovat. Jakmile je kompletní obsah stránky stažen, zbývá ho vložit na zásobník a označit jako návratovou hodnotu. Každý na zásobník vložený řetězec je okopírován, takže původní řetězec lze uvolnit. Na zásobník lze řetězec vložit dvěma způsoby. Buď není jeho délka uvedena a interpret ji spočítá sám, nebo lze délku explicitně specifikovat, čímž se vyhrazování místa v paměti pro kopii zrychlí. Jelikož bylo délku řetězce nutno průběžně ukládat, aby callback funkce mohla přidávat předané kusy webové stránky na konec řetězce bez zbytečných prodlev, je výsledná velikost k dispozici.

### 4.4.5 Funkce `postUrl` a `setReferer`

Druhou funkci tvořící jádro síťové komunikace JavaScript modulů představuje funkce `postUrl`. Značně se podobá té předchozí, v tomto případě však vykonává HTTP operaci POST. Jako jediná funkce přijímá tři parametry. První a druhý parametr je stejný jak u `getUrl`, tedy URL adresa webové stránky a pole, které obsahuje případné parametry k zakódování do předávané URL adresy. Poslední parametr je také pole, jenž obsahuje položky uspořádané stejným způsobem jako v případě parametru na druhé pozici. Jedná se o atributy a hodnoty, které mají být odeslané metodou POST. V případě této funkce musí být již druhý parametr uveden, i když není použit, kvůli třetímu parametru, který je vyžadován. Případ užití, kdy je zavolána funkce `postUrl`, aniž by jí byly předány POST parametry, nedává smysl, neboť právě přes tyto parametry jsou všechny potřebné informace spolu s textem SMS zprávy webové bráně zasílány. Knihovna libcurl přijímá POST parametry stejně zřetězené jako parametry vkládané do URL adresy. Ke změně formátu lze výhodně po menších úpravách využít funkci `appendParameters`. V tomto bodě je již vše připraveno k provedení POST požadavku. Nastane-li chyba kvůli síťové komunikaci, nedostatku paměti, lichému počtu prvků v některém z polí nebo prázdnému poli s POST parametry, jsou tyto chybové stavy výjimkou předány hlavnímu programu.

Poslední drobná funkce, která podle parametru nastavuje Referrer header v HTTP

požadavku při komunikaci s webovou bránou pravděpodobně pro statistické účely, nese název `setReferer`. K provedení operace je nutné mít ve funkci přístupný libcurl handle, neboť právě přes něj je Referrer header nastavován. Předávaný parametr opět není třeba zvlášť kontrolovat díky aplikaci vestavěné funkce `toString`. Došlo-li by při nastavování headeru k chybě, je chybový kód zachycen a poslán výjimkou ke zpracování.

### 4.4.6 Neimplementované funkce

V původní implementaci jsou dostupné navíc dvě další funkce, které nejsou v konzolové verzi programu přítomny. Funkce `recognizeImage` slouží k zobrazení CAPTCHA obrázku a jejím výsledkem je dekódovaný text, jenž uživatel zadá. Jelikož konzolový program není schopen obrázky zobrazit, nebudou se moduly s tímto zabezpečením používat a tak není třeba funkci implementovat. Druhá vynechaná funkce se nazývá `extractCountryPrefix`. Je součástí javovské třídy, která manipuluje s mezinárodními předvolbami. Třída obsahuje kromě dalších metod velké asociativní pole, ve kterém dvojice klíče a hodnoty představuje mezinárodní zkratku země a její odpovídající předvolbu. První komplikaci by představovalo použité asociativní pole. V jazyce C by bylo nutno tuto datovou strukturu simulovat dvěma poli a třetím mapovacím polem. Druhý a závažnější problém je objemnost pole s předvolbami, které nepatří k těm nejmenším. Vhodnější řešení je stejnou nebo podobnou funkci přesunout přímo do modulu, protože je funkce používaná pouze v jednom případě. Tímto bude funkce zabírat paměť jen v případě, kdy je konkrétní modul využíván, namísto trvalého zvětšení velikosti spustitelného souboru.

## 4.5 Zpřístupnění zadaných parametrů a rozšiřujících funkcí JavaScript interpretu

Všechny parametry zadané při spuštění programu je potřeba předat dovnitř modulu. Jelikož je hlavní modulová funkce `send` bezparametrická, k hodnotám parametrů se přistupuje stejně jako v případě rozšiřujících funkcí z globální úrovně JavaScript programu. Vložení proměnných musí proběhnout po inicializaci Duktape kontextu a před zavoláním funkce `send`, jestli proběhne před nebo po evaluaci samotného modulového souboru, není podstatné. Jako první krok je nutné na vrchol zásobníku umístit superglobální objekt. Poté je vždy na zásobník vložena hodnota parametru ve formě textového řetězce, která je přiřazena superglobálnímu objektu jako jeho patřičně pojmenovaný atribut. Proměnná `SENDERNAME` není jako jediná dostupná, neboť je potřebná jen v případě webových bran s CAPTCHA obrázky. V případě funkcí je tento proces trochu odlišný, neboť zahrnuje o jednu přiřazovací úroveň navíc. Na zásobníku je v této situaci pouze superglobální objekt. Posléze se vloží prázdný nepojmenovaný objekt, kterému jsou postupně přiřazovaný stejným postupem jednotlivé rozšiřující funkce jako pojmenované atributy. Všechny tyto funkce jsou v podstatě metody objektu `EXEC`, tudíž jako poslední krok je výsledný objekt přiřazen superglobálnímu objektu jako pojmenovaný atribut `EXEC`.

## 4.6 REST služba a volitelné funkce při překladu

Jako první je potřeba vytvořit část starající se o příchozí síťová spojení. Ta se skládá z typických konstrukcí k vytvoření TCP socketu, nabindování zvoleného portu z parametrů k socketu, naslouchání a akceptování příchozích spojení. Je-li přijato nové spojení, služba

standardně vytvoří pomocí volání `fork` nový proces, který požadavek klienta obslouží, a původní proces pokračuje v naslouchání na portu. Zde se nachází první volba při překladu, zda si uživatel vybere službu provozovat jako klasického démona, nebo ji plánuje zařadit do správy super-serverovému démonu. Hlavní rozdíl spočívá v kanálu, po kterém je výstup programu posílán – přes socket přímo vzdálenému klientovi nebo na standardní výstup, na který je napojený super-serverový démon. Větší část služby tvoří samotná obsluha obdrženého požadavku, kde jako první operaci je nutno provést dynamickou alokaci paměťových bloků pro příchozí a odchozí zprávu, bez kterých by požadavek nemohl být zpracován. Oba paměťové bloky jsou schopny uchovat maximálně 1023 textových znaků, nepočítáme-li s Unicode a jinými vícebajtovými způsoby kódování. Požadavek může pocházet z jednoduchého skriptu nebo z webového prohlížeče, kde je HTTP header podstatně rozsáhlejší. Velikost jedné SMS zprávy může podle GSM standardu být maximálně 1120 bitů[3]. Počet znaků ve zprávě se odvíjí od počtu bitů (7, 8 nebo 16) kódující jeden znak. Mnoho webových bran však zprávu dokáže rozdělit do více zpráv, je-li stanovený limit počtu znaků překročen, nebo ji jednoduše ořízne či skutečnost oznámí jako chybu. Je tedy brána v úvahu možná větší délka textu SMS zprávy v požadavku. Odpověď ze strany služby sestavuje pouze minimální HTTP header, webová brána však může poskytnout objemnější chybovou hlášku, kterou by výstupní paměťový blok měl byt schopen pojmout. Jakmile je požadavek uložen, začíná jeho zpracovávání. Nejdřív je z požadavku extrahována HTTP metoda a cesta ke zdroji a z obou je vytvořena dvouciferná konstanta identifikující operaci. Desítky označují HTTP metodu a jednotky zdroj. Zbytek obsluhy požadavku se odvíjí od hodnoty této vytvořené konstanty, díky čemuž lze schopnosti služby snadno rozšiřovat. Nepatří-li požadovaná operace mezi ty, které služba implementuje, je klientovi odeslán HTTP header se stavovým kódem „400 Bad Request“. Po identifikaci momentálně jediné operace je zbytek HTTP headeru přeskočen, neboť neobsahuje žádné další užitečné informace. Zde je při překladu možno povolit další funkci. Jedná se o kontrolu správnosti HTTP požadavku. Header je od těla požadavku oddělen 2x po sobě jdoucími CRLF znaky. Služba lokalizuje první výskyt těchto znaků a posune se ve zpracovávání za ně. Je možné zapnout kontrolu pro případ, že by se tyto znaky v požadavku nenacházely vůbec a tím by se nejednalo o validní HTTP požadavek, což by bylo vypsáno na standardní chybový výstup a obsluha by byla ukončena. Nyní může být proveden rozbor samotného obsahu požadavku v JSON formátu, o kterém se předpokládá, že je validní. Odpovídá-li momentálně zpracovávaný atribut některému z hledaných, je pozice jeho hodnoty zaznamenána do struktury, která simuluje proměnnou `argv`, přes kterou jsou standardně předávány parametry při startu programu. Po dokončení zpracování JSONu lze při překladu zapnout další kontrolu, zda jsou alespoň minimální potřebné parametry přítomny, tedy telefonní číslo příjemce, text SMS zprávy a název JavaScript modulu, a chybí-li některý z nich, odešle chybové oznámení. V případě, že by některý z těchto parametrů chyběl, dojde k havárii aktuálního obslužného procesu služby. Po zpracování požadavku a naplnění struktury s parametry je volána funkce, která spouští vlastní interpretaci modulu a odesílání SMS zprávy, která je identická s konzolovým programem. Jakmile je řízení z funkce vráceno, začne sestrojování odpovědi s výsledkem odesílání. HTTP header obsahuje pouze povinnou verzi HTTP protokolu a stavový kód, položku Content-Type a Content-Length. Content-Length je doplňována dodatečně, až je tělo odpovědi také ve formátu JSON složeno a tedy jeho délka známá. Zpráva obsahuje návratový kód interpretace udávající úspěch nebo typ chyby a všechny informativní a chybové hlášky, které byly komunikačními funkcemi z modulu předány, a které byly vyloženy z chyb samotné JavaScript interpretace.

# Kapitola 5

# Profiling

| Část z celkového času běhu | Počet invokací | Název funkce |
|---|---|---|
| 30,74 % | 8 505 660 | `duk_unicode_decode_xutf8.isra.0` |
| 13,13 % | 8 505 660 | `duk_unicode_decode_xutf8_checked` |
| 9,39 % | 761 874 | `duk__match_regexp'2` |

Tabulka 5.1: Výňatek ze statistiky o časově nejnáročnějších funkcích

Konzolový program byl pomocí nástroje callgrind[1] podroben profilování, které proběhlo na školním serveru Merlin[2]. K odeslání zprávy, která obsahovala „Profiling test“, byla použita webová brána Vodafone[3]. 98,45 % času života programu je stráveno ve funkci `duk_pcall` a funkcích v ní volaných. Jedná se o funkci Duktape interpretu, která invokuje zvolenou funkci ze strany JavaScriptu. V tabulce jsou uvedeny tři individuální funkce, ve kterých program strávil nejvíce času. Přes hierarchii volání funkcí lze zjistit, že jsou tyto funkce součástí procesu vyhledávání řetězce vyhovující zadanému regulárnímu výrazu. Operace match je aplikována na stažený obsah webové stránky, jejíž velikost může dosahovat 100 kB a více. Práce s regulárními výrazy tedy patří k nejsložitějším operacím, nacházejícím se v JavaScript modulech.

[1] http://valgrind.org
[2] http://merlin.fit.vutbr.cz
[3] https://park.vodafone.cz/egw/.0

# Kapitola 6

# Závěr

REST služba i konzolový program jsou plně funkční a reálně použitelné. Propojování jazyka C a JavaScript modulů probíhalo bez závažnějších problémů. Celková implementace byla přímočará a bez složitých konstrukcí. Doba odesílání SMS zprávy obvykle trvá 4 až 12 sekund, což je v přijatelných mezích, uvážíme-li, že obslužný modul nepoužívá žádné specializované API, ale ve své podstatě provádí téměř identické kroky jako uživatel při užívání webové brány skrz webový prohlížeč. Hlavní příčinou je používání regulárních výrazů s velkým objemem textových dat v JavaScript modulech. Zmenšením prohledávaného prostoru regulárními výrazy nebo změnou způsobu kontroly stavu přihlášení uživatele k webové bráně je možné program rapidně zrychlit. Poskytované schopnosti modulům lze snadno pro případ budoucích aktualizací rozšířit, stačí je jen naimplementovat a zpřístupnit skriptu. Stejně tomu tak je i u REST služby. Jsou zde nachystané funkce, které rozpoznávají poskytované operace, kam lze dodatečně požadované složky jednoduše doplnit, a tím je možné se plně soustředit na implementaci chování vytvářené části služby.

# Literatura


[1] History - GCC Wiki. [online], rev. 10. ledna 2008. [vid. 2015-05-17]. Dostupné z: https://gcc.gnu.org/wiki/History.

[2] Speaking UNIX: Working with inetd and xinetd, the Internet "super server". [online], vytvořeno 15. prosince 2009. [vid. 2015-05-17]. Dostupné z: http://www.ibm.com/developerworks/aix/library/au-spunix-inetd/.

[3] Groupe Spécial Mobile: Technical realization of the Short Message Service (SMS) Point-to-Point (PP) (GSM 03.40). Technická zpráva, European Telecommunications Standards Institute, červenec 1996.

[4] Hanák, D.: Stopařův průvodce REST API. [online], [vid. 2015-05-17]. Dostupné z: http://www.itnetwork.cz/stoparuv-pruvodce-rest-api.

[5] Shirali Shahreza, M.H., Shirali Shahreza, M.: An Anti-SMS-Spam Using CAPTCHA. In *Computing, Communications, Control, and Management, 2008. CCCM'08. ISECS International Colloquium on*, ročník 2., Guangzhou: IEEE, srpen 2008, s. 318–321, ISBN 978-0-7695-3290-5.

[6] Skonnard, A., Gudgin, M.: *XML - pohotová referenční příručka*. Grada Publishing, a.s., 2006, ISBN 80-247-0972-4.

# Příloha A

# Seznam rozšířujících funkcí pro JavaScript moduly

- `sleep(miliseconds)` – pozastaví proces na počet zvolených milisekund
  - `miliseconds` – čas v milisekundách
- `getUrl(url, parameters)` – pošle HTTP GET požadavek na webovou adresu a vrátí odpověď serveru
  - `url` – webová adresa stránky ve formé textového řetezce
  - `parameters` – pole se za sebou jdoucími dvojcemi atribut-klíč, které budou za URL adresu vloženy
- `postUrl(url, parameters, postData)` – pošle HTTP POST požadavek na webovou adresu spolu s POST parametry na webovou adresu a vrátí odpověď serveru
  - `url` – webová adresa stránky ve formé textového řetezce
  - `parameters` – pole se za sebou jdoucími dvojcemi atribut-klíč, které budou za URL adresu vloženy
  - `postData` – pole se za sebou jdoucími dvojcemi atribut-klíč, které budou metodou POST odeslány na adresu `url`
- `setReferer(referrer)` – nastaví HTTP header Referrer na požadovanou webovou adresu
  - `referrer` – URL adresa v textovém řetezci
- `setProblem(code, message)` – nastaví kód vzniklé chyby podle jejího názvu a vypíše případnou doplňující zprávu
  - `code` – textový řetezec reprezentující pojmenovanou hodnotu chyby
  - `message` – textový řetezec s detailem chyby; povinný jen pro určité kódy viz.
- `setSupplementalMessage(message)` – zobrazí uživateli zadanou informativní zprávu
  - `message` – textový řetezec s informativní zprávou

# Příloha B

# Seznam proměnných pro předávání parametrů do modulové funkce `send`

Množina potřebných proměnných se odvyjí od použitého modulu

- LOGIN – přihlašovací jméno uživatele k webové bráně
- MESSAGE – obsah SMS zprávy
- NUMBER – číslo mobilního telefonu příjemce
- PASSWORD – přihlašovací heslo uživatele k webové bráně
- SENDERNUMBER – telefonní číslo odesílatele

# Příloha C

# Seznam návratových a chybových kódů

- Chybové kódy pro modulovou funkci `setProblem` viz. výčtový datový typ `ProblemType_t` v hlavičkovém souboru `common.h`
- Návratové kódy konzolového programu a služby
  - `EXIT_SUCCESS` (0) – vše proběhlo v pořádku
  - `PARAMETER_ERROR` (1) – při zpracovávání parametrů došlo k chybě
  - `MODULE_ERROR` (2) – při evaluaci JavaScript modulu došlo k chybě
  - `CURL_ERROR` (3) – při síťové komunikaci nebo práci s knihovnou libcurl došlo k chybě
  - `DUKTAPE_ERROR` (4) – JavaScript interpret Duktape nahlásil chybu
  - `INTERNAL_ERROR` (5) – došlo k interní chybě (často při práci s pamětí)
  - `GATEWAY_ERROR` (6) – webová brána nahlásila chybu
  - `REQUEST_TOO_LONG` (7) – přichozí požadavek překročil limit 1023 znaků
- Návratové kódy specifické pro démona služby
  - `NO_PORT` (1) – nebyl zadán parametr číslo portu (v případě démonové verze)
  - `SOCKET_NOT_CREATED` (2) – socket se nepodařilo vytvořit
  - `BIND_ERROR` (3) – nebylo možno nabindovat zadaný port
  - `LISTEN_ERROR` (4) – nepodařilo se zahájit naslouchání na socketu

# Příloha D

# Formát JSON požadavku a odpovědi REST služby

- Požadavek je posílán metodou POST s adresou zdroje `/esmska/send`
- Atributy JSON požadavku (vše textové řetezce)
  - `NUMBER` – číslo mobilního telefonu příjemce
  - `USER` – přihlašovací jméno uživatele k webové bráně
  - `PASSWORD` – přihlašovací heslo uživatele k webové bráně
  - `GATEWAY` – nazév JavaScript modulu obsluhující webovou bránu
  - `TEXT` – obsah posílané SMS zprávy
  - `SENDERNUMBER` – telefonní číslo odesílatele
- Atributy JSON odpovědi
  - `RETCODE` – návratový kód interpretace (viz. návratové kódy konzolového programu a služby)
  - `ERROR` – chybová zpráva
  - `INFO` – informativní zpráva

# Příloha E

# Obsah CD

- esmska-cli – zdrojové soubory programu a služby
  - bin – spustitelné soubory a knihovna pro platformu Linux architektury x86-64
  - objects – meziprodukty překladu
  - src – zdrojové kódy programu a služby
    - cli – zdrojové kódy konzolového programu
    - include – hlavičkové soubory
    - libesmska – zdrojové kódy dynamické/statické knihovny
    - service – zdrojové kódy REST služby
  - INSTALL – stručný popis používání Makefile skriptu
  - LICENSE – licence programu v plném znění
  - Makefile – překladový a instalační skript
- thesis – soubory textu bakalářské práce
  - src – zdrojové LaTeX soubory práce
  - thesis.pdf – text práce ve formátu pdf