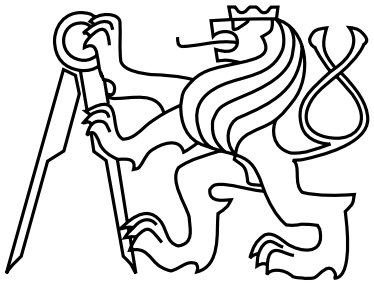
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Realizace e-mailové konference s vícejazyčnou podporou

Tomáš Čejka

Vedoucí práce: Ing. Josef Hlaváč

Studijní program: Elektrotechnika a informatika, dobíhající, Bakalářský

Obor: Výpočetní technika

26. května 2010

Poděkování

Touto formou bych chtěl poděkovat firmě Kassoft za poskytnutí přístupu na server.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 15. 5. 2010 .....

Abstract

The aim of this project is to build an e-mail conference, which could be used as an alternative service to the service Yahoo! Groups[23], that is nowadays used by the translator group. The assigment was created because of a reaction of users to their e-mails, which were sometimes delivered late and messages that were written in Czech were hardly readable due to mismatch of Charset Encoding.

The free software Mailman (GNU Mailing List Manager[20]) was chosen as a foundation. This is modular system, that supports extension of language set. The result of this project was built by editing the source code of Mailman.

Abstrakt

Cílem práce je vytvoření funkční e-mailové konference, která by sloužila uživatelům překladatelské skupiny jako alternativa k dosud využívané službě Yahoo! Groups[23]. Zadání práce vzniklo jako reakce uživatelů na občasné pomalé doručování zpráv a problémy se zobrazením zpráv psaných v češtině v různých kódováních.

Jako stavební kámen projektu byl vybrán volně dostupný software Mailman, the GNU Mailing List Manager[20], který vyhovuje jak z hlediska vícejazyčnosti, tak i modulárnosti, takže se dá do budoucna poměrně jednoduše rozšiřovat. Na tomto systému jsem vystavěl výsledný server e-mailové konference.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Tato práce byla iniciována skupinou překladatelů, která pro svou komunikaci a spolupráci až do této chvíle využívala služeb společnosti Yahoo!. Jedná se konkrétně o službu Yahoo! Groups[23], která se stará o rozesílání zpráv mezi účastníky, přihlášené uživatele do skupiny, a archivaci poslaných zpráv. Až doposud překladatelé provozovali svou e-mailovou konferenci právě díky této službě, jejíž archiv je pro registrované členy přístupný přes webový prohlížeč na serveru Yahoo!. Účastníci konference mohou své příspěvky přidávat prostřednictvím webového rozhraní, nebo přímo e-mailovou zprávou zaslanou na speciální adresu koference. Překladatelé s některými vlastnostmi nebyli příliš spokojeni, a proto jsem byl jsem osloven panem Ing. Josefem Hlaváčem, jedním z účastníků této konference. Služby Yahoo! Groups jsou sice komplexní a nabízí uživatelům komfortní užívání služeb, ale občas dochází ke značnému zpoždění doručování odeslaných zpráv, pravděpodobně kvůli vysoké zátěži serverů. Navíc se objevují i problémy se zobrazením českých příspěvků obsahujících znaky s diakritikou, což je pro překladatelskou konferenci nejzásadnější problém. Rozhodl jsem se, že se problémy pokusím vyřešit, a zaměřím se na objevující se nečitelnost některých zpráv v češtině. Vzhledem k velkému objemu dat a důležitosti přesnosti pochopení obsahu zpráv je nežádoucí, aby docházelo k podobnému zkomolení textu a tím i k narušení jejich sémantiky. Na základě domluvy jsem začal s přípravami na vytvoření nového serveru pro e-mailovou konferenci, který by po dobu testování fungoval paralelně s momentálně využívanou službou Yahoo! Groups. To znamená, že uživatelé by měli možnost přihlásit se do nové konference a dostávali by zprávy i z konference na Yahoo! Groups. Zároveň i příspěvky uživatelů nové konference by byly automaticky přeposílány do Yahoo! Groups. Tímto způsobem by měl být obsah obou konferencí stejný.

V tomto dokumentu bych chtěl popsat průběh realizace e-mailové konference, která by vyřešila zmíněné problémy. Práce začala výběrem vhodného software k použití, následovalo studium zdrojových kódů vybraného řešení, které bylo nakonec ještě nutné doplnit a poupravit tak, aby splňovalo všechny požadavky. V první řadě by nová konference měla umět rozpoznávat kódování, a v případě nutnosti kódování změnit tak, aby se zprávy uživatelům zobrazovaly správně. Protože software, který jsem vyhodnotil jako nejvhodnější, je napsán v programovacím jazyce Python[9], bylo nutné se s tímto jazykem seznámit, čemuž jsem věnoval značný podíl času práce na své bakalářské práci. Studium Pythonu probíhalo pročtením dokumentace, dostupné na internetu, zejména z oficiálních stránek Python Doc[10], konkrétně pak *Tutorial*. Po seznámení se se základy jazyka jsem prostudoval zdro-

jové kódy vybraného software, obzvláště pak sekce, které se zabývají zpracováním příchozí pošty a odesláním příspěvku mezi účastníky konference. V této fázi jsem zjistil, jak funguje většina částí software pro e-mailovou konferenci a souvisejících modulů. Činnost programu jsem se pokusil shrnout v jedné z následujících kapitol.

Na základě studia jsem mohl přistoupit k vlastní implementaci rozšiřujícího modulu pro rozpoznávání kódování textu každé příchozí zprávy (výsledkem je modul *Decode.py* více v kapitole Realizace na straně 11). Tato funkce je klíčová pro vyřešení problémů s nečitelností zpráv, což je hlavní problém, se kterým se překladatelé v současné době potýkají.

Po pokusu o rozpoznání bylo nutné zařídit, aby se příspěvky nové konference do Yahoo! Groups posílaly ve správném kódování. Správné kódování pak zajistí, že se na stránkách Yahoo! v archivu zobrazí text správně. Z tohoto důvodu jsem řešil jak vytvořit novou vlastnost systému - zakódování zpráv podle definovaného nastavení. Protože konference na Yahoo! Groups je z pohledu nové konference jedním z účastníků, bylo možné tuto funkci použít i pro ostatní uživatele. Nová konference poskytuje uživatelům možnost zvolit si znakovou sadu, ve které chtějí zprávy dostávat. Pro zajištění překódování zprávy během odesílání mezi účastníky nebylo potřeba vytvářet žádný další speciální modul, ale bylo nutné upravit stávající funkční kód, resp. doplnit ho o logiku zakódování podle zvolené znakové sady. Uživatelské rozhraní ani vlastní jádro systému konference neobsahovalo žádnou možnost uchování a využití uživatelské preference na volbu kódování, proto jsem musel vytvořit tuto funkci a přidat ji k ostatním uživatelským volbám. Pro tento účel byla upravena třída reprezentující e-mailovou konferenci, skripty generující webové stránky s uživatelským rozhraním nastavení, a nakonec modul, který se stará se o rozesílání zprávy mezi účastníky.

Kapitola 2

Popis problému, specifikace cíle

2.1 Úvod do problematiky

2.1.1 Fungování elektronické pošty

E-mailová korespondence patří v moderním světě k jednomu z nejčastějších komunikačních prostředků. Přenos e-mailových zpráv funguje na jednoduchém principu předávání textu resp. dat od odesílatele až k příjemci. Přestože široká veřejnost si již dávno zvykla na to, že po "odklikutí" zprávy tlačítkem Odeslat trvá jen okamžik, než se zpráva objeví příjemci ve složce "Přijaté zprávy", čas doručení ani vlastní úspěšnost doručení není nikde a nikým specifikována ani garantována. O odesílání zpráv se starají servery, komunikující přes SMTP (Simple Mail Transport Protocol)[7]. Samotné doručování zprávy probíhá traverzováním dat mezi nejméně dvěma poštovními servery. V poslední části cesty se zpráva přenese z e-mailové schránky do poštovního klienta, programu, který zprávu zobrazí adresátovi.

E-mailová zpráva se skládá z *hlaviček* a *těla* zprávy. V hlavičkách se uvádějí důležité údaje pro identifikaci zprávy a její doručení i informativní údaje o struktuře a obsahu zprávy. Pro nastínění struktury e-mailové zprávy je vhodné jednotlivé části popsat na vzorové zprávě. Tato vzorová zpráva vznikla vypuštěním některých částí z reálně přijaté zprávy. Pro zachování anonymity jsem přepsal údaje, které by mohly sloužit k identifikaci. Řetězec *[XXX.XXX.XXX.XXX]* obsahuje IP adresu stroje, který během předávání zprávy komunikuje.

```
Return-Path: <return@domainA.net>
Received: from domainA.net ([XXX.XXX.XXX.XXX]) by imap (Cyrus v2.2.13)
 with LMTPA; Thu, 16 Oct 2008 07:28:41 +0200
...
Subject: New Job Search
Reply-To: replyaddress@domainA.net
Sender: sender@domainA.net
Errors-To: errors@domainA.net
From: sender@domainA.net
Message-Id: <1763.20481016010933>
To: user@domainB.net
```

```
Precedence: normal
MIME-Version: 1.0
Date: Thu, 16 Oct 2008 01:18:16 -0400
Content-Type: multipart/alternative;
  boundary="------------Boundary-00=_0Y1AKPQYB"; charset=iso-8859-1
Content-Transfer-Encoding: 8bit
```

Podle zvyklostí každý poštovní server, na který se zpráva dostane, by měl přidat do hlavičky záznam Received. Za předpokladu, že tyto informace nebyly někde na cestě zfalšovány, jsme schopni zrekonstruovat cestu, kterou zpráva urazila. Hlavičky obsahují také důležité informace o odesílateli (From, Sender) a příjemci (To). Práci s organizací zpráv usnadňuje důležitý údaj - předmět zprávy (Subject)

Abychom mohli využívat možnosti formátování zprávy, nebo třeba připojovat ke zprávě přílohy, používá se pro e-mailovou korespondenci tzv. MIME formát[6]. MIME nabízí široké možnosti, většinou se ale používá jen několik z nich. Zprávy se dnes z většiny moderních poštovních klientů posílají jako text v HTML, jazyku pro tvorbu webových stránek. Aby se zajistilo, že obsah bude zobrazitelný i na počítačích s poštovním klientem, který HTML neumí zobrazit, posílají se zprávy s alternativní částí, která je čistě textová bez jakéhokoliv formátování. Takovýchto částí může být v jedné zprávě i více, každá však musí být označena oddělovačem *boundary*, jak je vidět v ukázce těla zprávy níže:

```
--------------Boundary-00=_0Y1AKPQYB
Content-Type: text/plain; charset=iso-8859-1
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

----------------------------------------------------------------
New Job Search
----------------------------------------------------------------

Dear Colleague

--------------Boundary-00=_0Y1AKPQYB
Content-Type: text/html; charset=iso-8859-1
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<HTML><HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>title</title></head>
<body bgcolor="#c0c0c0" TEXT="#000000" >
<div align="Left">
<font face="Arial" size="4"><span class="g-article_title">
New Job Search
```

```
</span></font>
<br>
<font face="verdana,arial" size="2">
<span class="g-article_content"><span id="article_synopsis">
<P><FONT face=Arial size=2>Dear Colleague</FONT></P></span><br>
</span></font></td></tr>
</div>
</body>
</html>
--------------Boundary-00=_0Y1AKPQYB--
```

Každá část zprávy v těle zprávy zpravidla začíná svojí hlavičkou. Hlavička označuje typ obsahu a způsob transformace dat. Transformace je nezbytná právě u přiložených souborů, kde se data zakódují do tisknutelných znaků (metodou *base64*), takže by nemělo dojít k poškození dat během přenosu (všechny poštovní servery by podle doporučení měly být schopny pracovat se zprávami obsahující pouze tisknutelné znaky ASCII). Z toho důvodu se používají transformace, obsahuje-li text neanglické znaky. Pro text zprávy je možné mimo *base64* použít tzv. *QuotedPrintable*.

2.1.2 Problémy s nevyžádanou poštou

Kvůli stále se množícímu výskytu tzv. SPAMu[22] dochází často k razantním opatřením na straně poskytovatelů e-mailových služeb. Proto se může stát, že zprávy jsou pozdrženy někdy i na desítky minut, jako je to v případě metody Greylistingu[3], která slouží jako účinná obrana proti SPAMu. Greylisting funguje na jednoduchém principu - poštovní server si při prvním pokusu o doručení příchozí zprávy uloží adresu odesílatele a zprávu v průběhu komunikace dočasně odmítne. Myšlenka se zakládá na faktu, že všechny plnohodnotné poštovní servery by se podle specifikací v RFC protokolu SMTP měli pokusit o opětovné odeslání. Při druhém přijímání, kdy server ví, že se odesílatel již v minulosti o předání zprávy pokusil, je už zpráva automaticky přijata a zařazena do schránky uživatele. Většina SPAMu přichází z počítačů, na kterých plnohodnotný SMTP server není spuštěn, a tak ke znovuodeslání již nedochází.

Pro účely e-mailové konference jsem si bohužel nemohl dovolit využít možností *greylistingu*, protože by to způsobovalo nežádoucí zpoždění doručování zpráv. Přesto bylo potřeba provést nějaké opatření proti SPAMu. Řešení představovala analýza textů zpráv antispamovým programem. Navíc e-mailová konference v základu používá metody tzv. *whitelistingu*, kdy zprávy jsou přijaty pouze od vyjmenovaných e-mailových adres, což v našem případě jsou adresy účastníků konference. Pokud do konference přijde zpráva, jejíž odesílatel není na seznamu uživatelů, je automaticky pozdržena a předána administrátorovi - moderátorovi konference ke schválení.

2.1.3 Požadavky na systém e-mailové konference

Činnost systému pro řízení a zajištění služby e-mailové konference by se dala shrnout do tří základních kroků. První fází je přijetí zprávy na server. Tato činnost je většinou zajištěna

přímo SMTP serverem, který kromě odesílání zpráv z fronty také poslouchá na TCP portu (většinou pro nešifrované spojení je to port 25) a čeká na příchozí spojení.

Druhá fáze obsahuje analýzu zprávy, kterou by již měl provádět systém konference. Analýzou se v tomto případě myslí kontrola odesílatele zprávy (protože zprávy, které nepochází od účastníků konference by se neměly rozeslat), úprava dat zprávy, a nakonec připravení verze k rozeslání mezi účastníky konference. Mezi úpravy zprávy patří většinou přidání nebo úprava hlaviček, jako například v předmětu zprávy se označuje, o kterou konferenci se jedná, do hlavičky Reply-To se zapisuje e-mailová adresa sloužící jako kontaktní adresa pro příchozí zprávy konference. V rámci řešení této práce bylo nutné upravovat i hlavičky, obsahující údaje o složení zprávy a použitém kódování textu.

Poslední fází průchodu zprávy systémem je samotné rozeslání zprávy mezi účastníky konference. V tomto okamžiku se naposledy mění hlavičky zprávy, konkrétně adresa doručitele, protože zpráva bývá systémem odesílána vždy každému účastníkovi zvlášť. O odeslání zprávy přes internet pomocí SMTP se stará opět poštovní server, který zprávu dostane do fronty odchozích zpráv. Pro zařazení zprávy do fronty je možné použít rovněž protokolu SMTP za předpokladu, že máme povolený *relay* zpráv z lokálního počítače (poštovní server i server konference je umístěný na stejném počítači).

2.2 Cíle projektu

Cíle tohoto projektu již byly nastíněny v předchozím textu. Projekt by měl řešit současné problémy, se kterými se potýkají překladatelé, účastníci e-mailové konference. Cíle by se daly zobecnit na dva úkoly. Prvním z nich je zřídit nový server s e-mailovou konferencí, který by mohl koexistovat se stávající konferencí na Yahoo! Groups. Od nového serveru se očekává rychlejší rozesílání zpráv od účastníků, protože zatížení bude mnohonásobně menší než na serverech Yahoo!.

Druhým úkolem, a také řešením největšího současného problému služby Yahoo! Groups, je vytvoření systému automatického rozpoznávání kódování. Z rozpoznaného kódování je potřeba zprávy překódovat do jednotné znakové sady, která bude definovaná pro celou konferenci. Definované kódování se dále musí změnit tak, aby se účastníkovi zpráva zobrazila správně. V první řadě je nutné zajistit, aby byly správně zobrazené zprávy v archivu na serveru Yahoo!, který bude v tuto chvíli zachován. Požadavek na čitelnost archivu bude splněn, jakmile budou zprávy přicházet na server Yahoo! v určitém konkrétním kódování.

Když bude zajištěno překódování zpráv do archivu na vhodné kódování, bude jednoduché zřídit seznam, ve kterém bude uložena nastavená znaková sada jednotlivých uživatelů. Podle nastavení pak bude zajištěno, že uživatel bude dostávat zprávu do své schránky v kódování, které si sám vybere nezávisle na kódování, ve kterém byla zpráva původně napsána. Nastavení by mělo být uživatelům přístupné přes webové rozhraní spolu s ostatními volbami uživatelů.

Na konci realizace by měl vzniknout nový server e-mailové konference dostupný na adrese *www.czechlist.org* s adresou *czechlist@czechlist.org* pro příchozí příspěvky uživatelů.

Kapitola 3

Analýza a návrh řešení

Na serveru, který jsem měl k dispozici a na kterém jsem měl možnost projekt realizovat, byla již před zahájením mé práce nainstalována linuxová distribuce Debian GNU/Linux[14]. Za účelem realizace e-mailové konference bylo v první řadě potřeba vybrat vhodný software, nebo se rozhodnout napsat vlastní zdrojový kód.

3.1 Vývoj konference

Vývoj vlastní aplikace, která by vyhovovala požadavkům e-mailové konference a navíc by vyřešila otázku rozpoznávání a dekódování znakové sady, je poměrně náročná varianta. Toto řešení by vyžadovalo napsat kompletní systém, který by uměl zpracovat příchozí poštu, udržovat seznamy účastníků konference, reagovat na nedoručené zprávy, generovat zprávy podle specifikací RFC. K problematice elektronické pošty a jejího doručování se váže mnoho specifikací, které by bylo nutné před psaním aplikace prostudovat. Ze strany uživatelů vyvstal požadavek na rychlost implementace, a protože by toto řešení bylo časově velmi náročné a nebylo by možné projekt včas dokončit, tuto variantu jsem zavrhl.

3.2 Existující řešení

Druhou možností bylo vyhledat již hotové řešení a to upravit podle požadavků překladatelů. Kromě vícejazyčnosti, rozšiřitelnosti a rychlosti manipulace se zprávami bylo nutné najít software, který bych mohl použít bezplatně. Proto jsem hledal opensource řešení, která jsou k dispozici zdarma ke stažení pro nekomerční účely. Na internetu jsem našel tyto produkty, ze kterých jsem vybíral:

1. phplist[2] (licence GPL, jazyk PHP)
2. Dada mail[13] (licence GNU General Public License, jazyk Perl)
3. GNU Mailman[20] (licence GNU General Public License, jazyk Python částečně C)

Přes internetovou encyklopedii Wikipedia jsem našel ještě dva produkty - LISTSERV a Majordomo, ale protože se jedná o proprietární produkty, neuvažoval jsem o jejich nasazení,

přestože se jedná podle popisu o bezproblémově fungující aplikace. Přistoupil jsem tedy k výběru jedné z uvedených aplikací.

3.2.1 phplist

Jako první jsem se seznámil s aplikací phplist. Jedná se o software napsaný v PHP, což je jazyk uživatelsky přívětivý a nenáročný. Protože jsem již v PHP programoval, zpočátku jsem považoval phplist za vhodného kandidáta pro realizaci projektu. Po instalaci a nastavení software jsem však zjistil, že se jedná spíše o evidenci e-mailových adres s možností odeslání hromadné zprávy. Phplist obsahuje webové rozhraní, přes které je možné napsat zprávu. Ta se následně může rozeslat mezi skupinu e-mailových adres. Adresy je možné filtrovat podle informací, které je možné k adrese přiřadit.

Software je vhodný spíše pro reklamní nebo obchodní účely, kdy je žádoucí jednu zprávu rozeslat hromadně mezi zaevidované účastníky, přičemž zpráva je určena pro zúžený okruh klientů, případně vybrat jen skupinu uživatelů, kteří tu samou zprávu ještě nedostali. Pro nasazení pro e-mailovou konferenci by bylo nezbytné implementovat část systému pro automatický příjem zpráv a jejich vyhodnocení, což je rozsáhlý úkol. Navíc by bylo potřeba vyřešit napojení této části na poštovní server, nebo alespoň použít doručování zpráv do uživatelské schránky a schránku potom pravidelně kontrolovat. Takové řešení by však nebylo příliš efektivní. Proto jsem se pokusil hledat dál.

3.2.2 Dada mail

Projekt Dada mail již podle popisu splňoval požadavky na konferenci. Sice v porovnání s proprietárním projektem Majordomo neumožňuje ovládání konference pomocí příkazů zaslaných v podobě e-mailové zprávy, ale disponuje webovým rozhraním, přes které se dá systém spravovat a ovládat. Jedinou nevýhodou pro mě byl programovací jazyk, ve kterém je systém napsaný - tedy jazyk Perl.

3.2.3 Mailman

Nakonec jsem se dostal k systému GNU Mailman. Tento produkt je robustní již na první pohled. Umožňuje provoz jedné a více e-mailových konferencí. Webové rozhraní i celý produkt je navržen pro snadnou lokalizaci, v základu obsahuje mnoho jazykových mutací (pro testovací účely jsem vybral tři - češtinu, angličtinu, němčinu). Kromě webového rozhraní je možné provádět základní akce konference, jako například přihlášení nebo odhlášení do konference, změna hesla, vypsání informací o konferenci nebo seznamu konferencí na serveru přímo prostřednictvím e-mailové zprávy odeslané na speciální adresu.

Pro administrátora konference systém nabízí webové rozhraní. Pro ovládání konferencí slouží i řada skriptů v Pythonu spustitelných z příkazové řádky. Systém je navržen tak, aby se úpravy nastavení daly jednoduše provést po nastudování zdrojových kódů i ručně, nebo si napsat vlastní modul pro administraci.

Přes webové rozhraní je dostupný archiv zpráv, který je podle konfigurace viditelný pro veřejnost, nebo jen pro registrované účastníky konference. Přestože pro účely tohoto projektu

se s využitím archivu nepočítá, spíše se zachová archiv na server Yahoo!, do budoucna tato funkce nabízí různá rozšíření, jako například automatické generování glosáře.

Mailman je psaný převážně v jazyce Python, pouze některé části, které slouží k napojení na webový server nebo na poštovní server jsou napsané v jazyce C.

3.3 Výběr řešení

Rozhodoval jsem tedy mezi projekty Dada mail a GNU Mailman. Po krátkém seznámení s oběma programovacími jazyky, v nichž jsou projekty napsány (Perl, Python), jsem vybral projekt Mailman pro realizaci konference a začal jsem studovat programovací jazyk Python, abych lépe porozuměl zdrojovým kódům programu a orientoval se v jejich činnosti. Čerpal jsem převážně z internetu, kde je velké množství návodů a výukových článků pro tento velmi oblíbený a populární programovací jazyk. Mezi hlavní zdroje bych zařadil oficiální stránky projektu[9], na kterých lze najít všechny podstatné informace pro seznámení se s jazykem.

3.4 Seznámení se s jazykem Python

Python je programovací jazyk, který bych přirovnal k bashi, ve kterém jsem již pracoval. V Pythonu je možné napsat téměř vše v relativně krátkém čase. Python je rozšiřitelný o moduly, napsané například v jazyce C, které se mohou po naimportování v programech používat. David M. Beazley v přednášce na Open Source Conference (17.červenec 2000)[1] uvedl jednoduchý přiklad, jak vytvořit modul pro výpočet největšího společného dělitele tak, aby se dal jednoduše volat v Pythonu. Příklad byl napsán v jazyce C. Slovy autora “It’s almost too easy...”.

Zjistil jsem, že Python je ideální jazyk pro programátory, kteří se chtějí soustředit spíše na sémantiku než syntaxi. Syntaxe je jednoduchá a řekl bych, že začínající programátoři si na ni rychle zvyknou. Oproti ostatním jazykům, se kterými jsem se seznámil, Python má zcela odlišný přístup k seskupování příkazů do bloků. Tam kde jazyk C a jiné podobné jazyky použijí závorky, které ohraničují bloky kódu, Python používá prosté odsazování řádků kódu. Řádky, které jsou napsané pod sebou, patří do stejného bloku právě tehdy, když jsou stejně odsazeny (řádky začínají stejným počtem *bílých znaků*). Pokud je zdrojový kód psaný kvalitně, tedy neobsahuje žádné příliš dlouhé bloky, je zdrojový kód v Pythonu přehledný a programátor si ušetří otevírací a uzavírací závorku.

Myslím, že velmi silnou stránkou Pythonu jsou jeho seznamy a slovníky a jednoduchá práce s nimi. U všech datových struktur, které jsou součástí základní sady modulů dostupných po instalaci, je možné procházení pomocí *iterátoru*. Dokonce i procházení textového řetězce po jednotlivých znacích se dá provést standardním způsobem:

```
for znak in retezec:
   print znak
```

Příklad vypíše jednotlivé znaky řetězce. Podobná syntaxe cyklu *for* se seznamem prvků existuje například v jazyce bash, PHP disponuje syntaxí s klíčovým slovem *foreach*.

Výhodou Pythonu může být i správa paměti. Protože se jazyk používá jako skriptovací, o správu paměti se stará interpret Pythonu a programátor zůstává ušetřen těchto starostí.

Během studia jazyka jsem však také narazil na jednu nevýhodu. Porozumění zdrojovému kódu je ztíženo tím, že typová kontrola Pythonu je poměrně benevolentní. Podobně k této problematice přistupuje PHP nebo JavaScript. Proměnné mají datový typ podle hodnoty nebo instance objektu, která se do nich zapíše. Proto i argumenty funkcí nejsou pevně dány resp. na první pohled nemusí být čtenáři kódu jasné, jakého typu argumenty jsou. Toto je výrazný rozdíl oproti jazykům jako je například *Java*, kde typová kontrola striktně probíhá již během překladu, ale i na úrovni tzv. *Java assembleru*. V Pythonu pokud dojde k situaci, kdy se pracuje se dvěma navzájem nekompatibilními typy, interpret vyhodí pouze výjimku, ale po jejím odchycení může vykonávání skriptu pokračovat dále.

Tuto nevýhodu ale vyvažuje možnost ladění skriptů, které umožňuje zjištění obsahu a typu za běhu programu. V průběhu své práce jsem používal ladící nástroj *pdb*[8], dodávaný spolu s Pythonem. Tak jako asi všechny ladící nástroje typu *gdb*, *pdb* umožňuje během krokování laděného procesu zobrazit hodnoty proměnných. Pdb navíc ale umožňuje přímo vykonávat libovolné přikazy tak, jako by byly součástí laděného skriptu. V porovnání s gdb a použitím jeho funkce *call* je použití v pdb mnohem jednodušší. Krokování v pdb dovoluje programátorovi i zajímavý příkaz *jump*. Tímto způsobem se mi povedlo například skočit zpátky v programu a tím opakovat zkoumanou metodu, což mi ušetřilo spoustu času.

Snad jediné, co mi při používání pdb nebo i interpretu Pythonu chybělo, byla možnost uložení vykonaných příkazů, které jsem během ladění nebo práce v interpretu zadal. Myslím, že by to ulehčilo práci při vytváření nebo úpravě skriptu, který je zrovna laděn.

Kapitola 4

Realizace

4.1 Jak funguje systém Mailman

Tato kapitola pojednává o instalaci aplikace Mailman a její struktuře. Začátek kapitoly by měl shrnout postup zprovoznění konference a souvisejících částí systému. Strukturu jsem zjistil v průběhu studování zdrojových kódů systému. Během své práce jsem pomocí ladícího nástroje *pdb* odkrokoval většinu kódu, která se zabývá příjímáním zpráv do konference a jejich zpracováním a odesláním na jednotlivé adresy účastníků. Na základě studia kódu se pokusím stručně vysvětlit, jak probíhá proces průchodu zprávy konferencí. Po tomto úvodu dále nastíním, jak jsem postupoval při implementaci modulu na automatické rozeznávání kódování zpráv a jakým způsobem jsem zařídil převod textu zprávy na kódování, které si může uživatel sám zvolit.

4.1.1 Instalace a konfigurace

Jako poštovní server byl zvolen systém *Exim 4*[5] spolu s antispamovým řešením a antivirem. Instalace byla provedena z repozitáře.

```
 apt-get install exim4-daemon-heavy spamassassin sa-exim clamav \
clamav-base clamav-daemon clamav-data
```

Tímto příkazem se nainstaluje systém Exim ve verzi, která umožňuje přidání filtrování zpráv, čehož využívá antispamový filtr *SpamAssassin*. Vybraným antivirovým scannerem byl rozšířený produkt *ClamAV*. V distribuci Debian je možné využít poloautomatické základní konfigurace přes program *dpkg-reconfigure*. Exim má předpřipravené skripty, které je možné pomocí dpkg-reconfigure spustit:

```
 dpkg-reconfigure exim4-config
```

Po instalaci je poštovní server připravený k základnímu použití. Dále se budu zabývat pouze instalací a konfigurací, která přímo souvisí se systémem Mailman.

Mailman je rozšířená aplikace ve světě e-mailových konferencí, proto je možné ji najít v oficiálních repozitářích linuxových distribucí. Díky tomu instalace systému probíhala

snadno, protože distribuce Debian v repozitáři Mailman nabízí také. Z toho důvodu jsem nemusel instalovat aplikaci přímo ze zdrojových kódů, což by mělo za následek kompilování programů v jazyce C, které Mailman ke své práci používá. Tak jako všechny aplikace z repozitáře, systém Mailman lze nainstalovat pomocí balíčkovacího systému Debianu příkazem *apt-get* s vhodnými parametry.

```
apt-get install mailman
```

Podle návodu nalezeného na internetu[4] jsem nakonfiguroval soubor aliasů e-mailových adres tak, aby poštovní server předával příchozí poštu určenou speciální adrese přímo Mailmanu. Tuto funkčnost zajišťuje konfigurační soubor *aliases*. Na použitém stroji je celá cesta k souboru */etc/aliases*. Soubor se následně přeloží příkazem *newaliases*, který zaktualizuje databázi lokálně doručovaných e-mailových adres.

Pro správné fungování práce poštovního serveru se zprávami bylo dále potřeba poupravit některé konfigurační soubory. Úprava spočívala v editaci sekcí *Transport* a *Router* podle návodu na oficiálních stránkách projektu exim v kapitole HowTo[17]. Po modifikaci konfiguračních souborů a znovuspuštění poštovního serveru se objevila chyba během přijímání zpráv do konference. Zprávy byly zahazovány a poštovní server hlásil chyby, které znamenaly, že server nezná cílovou službu doručení. Problém jsem vyhledal v logu poštovního serveru a zjistil jsem, že příčina byla ve výchozím nastavení v sekci typů doručení. Výchozí nastavení totiž nemělo povolené doručování do tzv. *PIPE*, čehož ale Mailman využívá. Do hlavního konfiguračního souboru eximu jsem přidal makro, které tento způsob povolovalo a server začal po dalším načtení konfigurace zprávy přijímat a přes PIPEu doručovat wrapperu Mailmanu.

Mailman disponuje webovým rozhraním, které zahrnuje správu konferencí, rozhraní pro uživatelské nastavení a archiv zpráv. Aby bylo webové rozhraní přístupné a funkční, je potřeba nastavit webový server. O instalaci webového serveru jsem se nemusel starat, protože jsem měl k dispozici webový server Apache[15].

Musel jsem však přidat do konfigurace stránky Mailmanu. Rozhodl jsem se nakonfigurovat nový virtuální webový server, protože s jeho vytvářením jsem již měl zkušenosti. Konfigurace je komplexní a jednoduše přenositelná - celé nastavení má svoji sekci. Hostování více webových stránek umožňuje protokol HTTP 1.1. Webový server rozpoznává, který virtuální server má načíst podle údaje z hlavičky dotazu, konkrétně se jedná o hlavičku *Host*. Pro tento účel je důležitý atribut ServerName udávající, na které doméně má webová služba Mailmanu běžet. Příklad HTTP dotazu verze 1.1:

```
GET / HTTP/1.1
Host: www.czechlist.org
```

Zbývalo jen podle téhož návodu[4] jako při konfiguraci aliasů nastavit požadované cesty ke skriptům. Adresy webového adresáře jsem změnil tak, aby souhlasily s požadovaným umístěním na serveru stránek. Konfiguraci Apache jsem uvedl v dodatku na straně 33.

Z uživatelského hlediska jsem provedl přes webové rozhraní Mailmanu pár nastavení, jako například zvolení jména e-mailové konference, a konference byla zprovozněna.

4.1.2 Činnost částí systému

Pro splnění dalších bodů projektu bylo nutné prostudovat fungování systému a na základě toho rozhodnout o nejvhodnějším místě doplnění požadovaných modulů na rozpoznávání kódování a zakódování zprávy. Po důkladném prostudování kódu přišla na řadu implementace. Tato podsekce popisuje strukturu systému Mailman a popisuje, jakým způsobem zprávy procházejí skrz jednotlivé moduly, které se starají o zpracování zpráv.

Po zprovoznění poštovního serveru jsem zkoumal napojení na Mailman. Jak již bylo řečeno, příchozí zpráva je poslána přes PIPEu přednastavenému programu na standardní vstup. Tímto programem je wrapper, napsaný v jazyce C, který je součástí Mailmanu. Program nedělá nic jiného, než jen že zprávu předá skriptu v Pythonu, a ten ji potom uloží do speciálního adresáře *in* pro příchozí poštu. Před uložením však ještě provede jakési zaobalení a výsledný soubor uloží v textovém formátu pro serializaci objektů Pythonu, tzv. *pickle*. Jak jsem zjistil, pickle je zavedený způsob serializace zpráv v Mailmanu, a v tomto formátu jsou ukládány všechny objekty Mailmanu, které je potřeba k nějakému účelu zachovat.

Kromě adresáře *in* využívá Mailman ještě další: *archive*, *bad*, *bounces*, *commands*, *news*, *out*, *retry*, *shunt*, *virgin*. Pro práci se soubory v těchto adresářích je spuštěno několik procesů, tzv. *Runnerů*. Každý z těchto Runnerů je skript, obsahující třídu, zděděnou od třídy Runner. Jedná se například o *IncomingRunner*, který se stará o celé zpracování příchozích zpráv, wrapperem uložených do adresáře *in*. Zpracovávání IncomingRunnerem bude detailněji probráno níže. Dále bych uvedl ještě OutgoingRunner, CommandRunner a VirginRunner. OutgoingRunner odesílá zprávy zpracované od IncomingRunneru. Pro rozeslání zprávy OutgoingRunner volá metodu třídy SMTPDirect, která přímo navazuje spojení s lokálním SMTP serverem. OutgoingRunner, resp. SMTPDirect jsem upravoval za účelem překódování zprávy na uživatelem preferované kódování. CommandRunner slouží k obsluze uživatelských příkazů, které přicházejí ve formě e-mailové zprávy. VirginRunner se stará o zprávy generované samotným Mailmanem. Svou speciální frontu, a tím i Runner, má také archiv zpráv, zde se jedná o ArchRunner.

Každý z Runnerů v nekonečné smyčce kontroluje, zda nemá ve své frontě nějakou zprávu ke zpracování. Pokud žádné zprávy nejsou, čeká se než nějaká přijde. Ve chvíli, kdy se objeví ve frontě nějaká zpráva, činnost se liší mezi jednotlivými frontami. Blíže popíši IncomingRunner, protože je jeho činnost stěžejní pro úkol rozpoznávání kódování. V zásadě všechny Runnery volají odpovídající Handler nebo Handlery, kterým zprávu postupně předávají. Což v podstatě odpovídá návrhovému vzoru Strategy, kdy se změnou volané instance třídy mění algoritmus zpracování dat.

4.1.3 Průchod e-mailové zprávy systémem

Každá zpráva, která prochází systémem, má svoji vlastní datovou strukturu *msgdata*. V této struktuře jsou uloženy všechny potřebné údaje o zprávě. Když se objeví nějaká zpráva v adresáři *in* a IncomingRunner je spuštěn, začne se zpráva zpracovávat.

Na začátku tohoto procesu se do msgdata přidá seznam *pipeline*, což je seznam Handlerů, které jsou postupně volány. IncomingRunner projde v cyklu *pipeline*, každý modul během iterace importuje a zavolá jeho metodu *process*, které se předává zpráva spolu s msgdata

jako parametry. Tímto způsobem je zaručena modularita systému, protože do *pipeline* je možné přidat libovolný modul, který má implementovanou metodu *process*.

Seznam Handlerů, které se zařazují do pipeline pro příchozí zprávy, je definován v konfiguračním skriptu *Defaults.py* v proměnné *GLOBAL_PIPELINE* spolu s ostatními důležitými volbami. Jako hlavní konfigurační soubor Mailmanu se uvádí soubor *mm_cfg.py*, ale ten importuje právě soubor Defaults.py, kde jsem našel většinu nastavených proměnných. Proto po implementaci vlastního modulu stačilo doplnit jeho jméno do seznamu v proměnné GLOBAL_PIPELINE a Mailman začal nový modul volat vedle původních Handlerů.

Během své práce jsem prošel všechny Handlery ve frontě příchozích zpráv. Jedná se o tyto Handlery (u některých připojuji krátký popis):

1. SpamDetect
2. Approve
3. Replybot
4. Moderate
   Tento modul byl důležitý při napojování nové e-mailové konference na Yahoo! Groups, protože pokud odesílatel není v seznamu účastníků, zpráva se mezi platné účastníky konference nerozešle. Proto jsem musel provést drobnou změnu tak, aby se přijaly všechny zprávy uživatelů, kteří jsou členy Yahoo! Groups. Zjistil jsem, že Yahoo! nastavuje hlavičku *Reply-To* tak, aby odpovědi uživatelů na příspěvky byly směrovány zpět do konference místo odesílateli příspěvku. Problém jsem vyřešil tak, že v hlavičce zprávy měním údaje o odesílateli na adresu z *Reply-To*. Tím jsem "zastřešil" všechny uživatele Yahoo! Groups a zabezpečil příjem zpráv z této konference.
5. Hold
   Kontrola délky zprávy kvůli případnému pozdržení; tento modul by měl zabránit rozesílání hodně dlouhých zpráv; když se objeví dlouhá zpráva, musí být potvrzena moderátorem
6. MimeDel
   Tento modul jsem aktivoval nastavením *filter_content*; modul odstraňuje alternativní části zprávy ve formátu MIME, zůstane jen část text/plain, což zjednodušuje rozpoznávání kódování.
7. Decode
   Nový modul - bude popsáno v samostatném odstavci.
8. Scrubber
9. Emergency
   Pokud je konference ve stavu poruchy, zprávy, které nejsou schválené moderátorem, se nedoručují.
10. Tagger

11. CalcRecips
    Vytvoření seznamu příjemců (účastníků konference) a aktualizace tohoto seznamu v msgdata.

12. AvoidDuplicates

13. Cleanse
    Modul se stará o hromadné mazání hlaviček. Do tohoto modulu jsem přidal mazání hlaviček, které přidává Yahoo!, a záznamy Received. Sice jsme tím ztratili informace o cestě zprávy, ale pro rozeslání zprávy mezi účastníky tato informace není důležitá.

14. CleanseDKIM
    Odstraňuje některé hlavičky zprávy konkrétně DKIM, tedy tzv DomainKeys identified mail[16]. Tento modul jsem zapnul nastavením proměnné
    REMOVE_DKIM_HEADERS na hodnotu *yes*, protože jsem se snažil o co největší minimalizaci zprávy.

15. CookHeaders
    Upravuje hlavičky zprávy, například předmět - v případě prázdného předmětu se vyplní *(no subject)*.

16. ToDigest
    Kopie zprávy se zapisuje do */var/lib/mailman/lists/czechlist/digest.mbox*.

17. ToArchive
    Zařazení kopie zprávy do archivu.

18. ToUsenet
    Nepoužívá se, ale Mailman nabízí funkci brány k news serveru.

19. AfterDelivery
    Inkrementuje počet odeslaných zpráv konference a nastaví proměnnou s časem posledního odeslání zprávy. Nezjistil jsem, k čemu se tyto hodnoty dále používají.

20. Acknowledge
    Pokud má uživatel v nastavení, že chce obdržet potvrzení o doručení, tento modul se postará o vygenerování příslušné zprávy.

21. ToOutgoing
    Zapsání zprávy do odchozí fronty, tedy do adresáře *out*.

Handlery jsem uvedl ve stejném pořadí, jako jsou volány za sebou. Pokud se má z nějakého důvodu odesílání zprávy v kterémkoliv bodě zpracování přerušit, stačí v příslušném Handleru vyhodit výjimku, která způsobí ukončení provádění *pipeline*. Všechny chyby nebo přerušení, které se za běhu Mailmanu objeví, se zaznamenávají do systémového logu */var/log/mailman/error*. Zároveň se ukládájí i záznamy o úspěšně přijatých zprávách i o adresách, na které se zprávy rozeslaly, do souborů
/var/log/mailman/post a */var/log/mailman/smtp*.

4.2 Implementace rozpoznávání kódování zpráv

V této sekci následuje vlastní popis realizace rozpoznávání kódování a implementace modulu Decode.py. Pro přiblížení jsem shrnul problematiku znakových sad a snažil se identifikovat příčiny problému se špatným zobrazením znaků s diakritikou. Pro svou práci jsem používal mechanismy Pythonu pro manipulaci s různými znakovými sadami a jejich převody. Konec sekce pojednává o tom, jak funguje modul Decode.py.

4.2.1 Problematika znakových sad

Určení znakové sady je poměrně složitá úloha, protože se vlastně jedná o určení způsobu, kterým se mají interpretovat data. Znaková sada neboli kódování nám říká, jaký význam se má přiřadit jednomu nebo skupině bajtů. Znakových sad existuje mnoho. Některé jsou jednobajtové (např. ASCII), jiné jsou vícebajtové (např. UNICODE). Existují i varianty, které kombinují obě varianty za účelem snížení objemu dat (např. UTF-8).

UTF-8 je dnes velmi rozšířené kódování, používá se například na webových stránkách a je výchozím kódováním většiny současných linuxových distribucí. Jeho velkou výhodou je, že stejně jako UNICODE obsahuje všechny znaky všech existujících písem. Kromě toho nezabírá každý znak nutně 4 bajty, jako je tomu u UNICODE. Tato vlastnost může být i nevýhodou, protože se například složitějším způsobem určuje skutečný počet znaků textového řetězce (u znakových sad s konstantní velikostí jednoho znaku stačí celkovou velikost textového řetězce v bajtech vydělit velikostí jednoho znaku, toto pravidlo však obecně pro UTF-8 neplatí).

Z toho důvodu když budeme interpretovat text napsaný v kódování UTF-8 stejně, jako by se jednalo o libovolné kódování s konstantní velikostí znaků, dostaneme text, který se bude od originálu lišit minimálně počtem znaků. Problém samozřejmě vzniká i opačnou interpretací, ale protože základní množina znaků má většinou stejnou hodnotu v různých kódováních, text je čitelný. Pokud například použijeme jen znaky z ASCII tabulky s hodnotami do 0x7E, což odpovídá anglické abecedě, neměl by se problém objevit.

4.2.2 Příčina problémů s kódováním

Příčina problému, se kterým se skupina překladatelů potýká, tedy spočívá v chybné interpretaci textů zpráv. Tím, že se v ekosystému internetu objevuje spousta různých programů, a dokonce i mnoho různých verzí téhož programu, koexituje vedle sebe i mnoho kódování. Během analýzy zpráv, které se posílají do konference jsem zjistil, že ne všechny programy sloužící jako poštovní klient zacházejí s textem zprávy a obecně s daty zprávy správně. Například jsem přišel na to, že občas vzniká problém s kódováním textu zprávy, který vznikne jako odpověď na nějakou předchozí zprávu. Problémový poštovní klient při citování pravděpodobně vzal text původní zprávy, ten vložil do nové zprávy, kam uživatel dále připsal vlastní text. Bohužel ale nebylo zajištěno, aby byl původní text ve stejném kódování jako nový text.

Kromě uvedeného problému se navíc stávalo, že údaj v hlavičce zprávy o použitém kódování buď úplně chyběl, nebo byl nepřesný - v hlavičce jedné části zprávy může být jen jedno

kódování, což nutně vede k nepravdivosti v případě, že zpráva obsahuje dva texty v různých kódováních. U takového textu se pak v praxi zobrazí některá část výsledné zprávy nečitelně. Chyba samozřejmě není na straně serveru, ale spíše na straně poštovních klientů používaných uživateli. Přesto jsem se pokusil tento problém nějak řešit. Předně pokud by se nám povedlo zařídit, aby uživatelé z konference dostávali zprávy vždy ve stejném, předem zvoleném kódování, nemohl by vzniknout problém s citováním textu předchozí zprávy. Zároveň by se tím minimalizovalo riziko výskytu chybného údaje o kódování zprávy v hlavičce zprávy.

Při řešení rozpoznávání kódování jsem vycházel z faktu, že uživatelé v dnešní době používají omezenou množinu znakových sad. Proto je výběr o něco jednodušší - testuji, zda je text v nějakém kódování z této množiny. Kromě testů kódování aplikovaných na zprávu jsem se rozhodl použít již existující řešení, které by mi rozhodování usnadnilo. Takovým řešením je pouštět program *enca*[24]. Jedná se o analýzátor dat/textu, který detekuje kódování na základě znalosti jazyka, ve kterém text je, a zákonitostí, které v daném jazyce platí. Enca podporuje jazyky střední a východní Evropy, takže pro rozpoznávání českých a slovenských příspěvků je možné tento program použít. V průběhu testování jsem zjistil, že není možné se na program enca spolehnout na sto procent. Někdy se totiž stává, že enca neumí o kódování rozhodnout. Na druhou stranu se po dobu testů nestalo, že by vrátil chybné kódování.

4.2.3 Práce s kódováním v jazyce Python

Abych mohl popsat činnost modulu na rozpoznávání kódování, měl bych se zmínit o tom, jak se s různými znakovými sadami pracuje v jazyce Python. Python obsahuje modul *unicode*, který dokáže překódovat vstupní data z uvedeného kódování na UNICODE. Při vytváření instance třídy konstruktor unicode umožňuje zadat jako parametr znakovou sadu vstupních dat, ze kterých se má instance vytvořit. Práce s kódováním je uživatelsky přívětivá, protože ve výsledku stačí pouze vstupní text převést do UNICODE při znalosti kódování textu. Ve chvíli, kdy je potřeba text zakódovat do nějaké jiné znakové sady, se zavolá metoda *encode* s parametrem výsledného kódování. Postup při použití třídy unicode demonstruje následující příklad:

```
>>> a~= unicode( "český text v~utf-8", "utf-8" )
#momentálně je v~proměnné a~text v~unicode
>>> b = a.encode( "iso-8859-2" )
#do b se přiřadil text v~iso-8859-2
```

Příklad ukazuje, jak je možné text v UTF-8 jednoduše převést na kódování ISO-8859-2.

4.2.4 Spouštění externího programu pro rozpoznávání

V tomto odstavci se budu věnovat modulu Decode.py, který využívá vlastností programu enca a třídy unicode v jazyce Python. Python nabízí modul *subprocess*, jehož metoda *Popen* dovoluje spustit externí program a komunikovat s ním prostřednictvím *pipe*y. Modulu subprocess jsem využil ve spojitosti s nasazením programu enca.

```
>>> payloadenc = Popen(["enca", "-g", "-i", "-"],\
   stdin=PIPE, stdout=PIPE).communicate(input=payload)[0]
```

Toto je příklad použití Popen, ve kterém pracuji se standardním vstupem a výstupem programu enca, spuštěného se seznamem parametrů *-g*, *-i*, -. Parametry říkají programu, že se má pokusit uhádnout kódování (*-g*), že má vzít vstupní data ze standardního vstupu (-) a poslední parametr (*-i*) nastavuje formát výstupního textu - jedná se o název kódování.

Konkrétně parametr *-i* zajistí jména kódování stejná jako používá program *iconv*, což je program, jehož funkce (iconv_open, iconv, iconv_close) jsou obsaženy v knihovně *libc*.

Na vstup programu se pošle text předaný parametrem input metody communicate. Zároveň metoda communicate vrací tzv. *tuple*. Indexem *0* dostaneme hodnotu - text z výstupu programu.

4.2.5 Nový modul - Decode.py

Modul Decode.py nejprve dekóduje hlavičky - položky *From* a *Subject*, které mohou obsahovat texty s neanglickými znaky. Tyto položky je nutné na konci procesu zakódovat do jednotného kódování (zvolil jsem UTF-8).

Dále následuje test, zda je zpráva složena z více částí. Tato situace by po průchodu modulem MIMEDel neměla nastat, proto tato větev podmínky neprovádí žádný test. Pokud zpráva obsahuje jen jednu část, zkopíruje se obsah zprávy a spustí se test kódování programem enca. Program enca se spouší pomocí modulu subprocess. Pokud enca nevrátí '???', což znamená, že nebylo možné rozhodnout o kódování, považuji výsledek za směrodatný. V opačném případě se musím pokusit uhádnout kódování jinak.

Využívám k tomu konstruktoru třídy unicode a pokouším se text překódovat do UNICODE. Pomocí testovací věty, která obsahovala všechna česká písmena s diakritikou, jsem zjistil, že každý pokus o rozkódování textu v libovolném kódování (kromě ASCII samozřejmě) selže, pokud se snažím text rozkódovat jako by byl napsaný v UTF-8. Při tomto testu navíc v konstruktoru třídy unicode používám parametr *strict*, který by měl zabezpečit okamžité vyhození výjimky, jakmile se při dekódování objeví sekvence, kterou kódování nemůže obsahovat. Kódování UTF-8 má totiž u vícebajtových znaků pevně danou strukturu jednotlivých bajtů.

Když se povede převod z UTF-8, uvažuji toto kódování jako vhodného kandidáta a další test přeskočím, protože již není dostatečně přesný. Jinak testování pokračuje vyzkoušením množiny obvyklých kódování. Přesnost a jistota rozhodnutí je menší, protože znakové sady typu CP1250 a ISO-8859-2 jsou podobné, a pokus o rozkódování většinou žádnou výjimku nevyhazuje, přestože výsledný text není v pořádku. Abych zohlednil informaci o kódování z hlaviček zprávy, přidávám do množiny na první místo toto kódování, pokud v hlavičce vůbec nějaké uvedeno je. Protože se v tuto chvíli již nedá s jistotou rozhodnout na základě výsledků konverze pomocí modulu unicode, použiji první úspěšné kódování. O možných úpravách tohoto řešení se zmíním v závěru.

Nakonec ještě překóduji text do UTF-8, centrálního kódování, ve kterém v systému ukládám zprávy. Pokud byl před vstupem do Decode modulu v hlavičce údaj o kódování a tento údaj se lišil od rozpoznaného, resp. uhádnutého kódování, přidám do hlavičky informaci o konverzi - ze kterého a do kterého kódování se konverze prováděla.

U překódování hlaviček From a Subject používám pouze program enca. Poslední fází je modifikace instance zprávy, tedy nahrazení původního textu překódovaným, nahrazení

hlaviček *From* a *Subject*. Původní hlavičky je samozřejmě nutné nejprve smazat, aby nedošlo k nechtěné duplikaci těchto údajů.

4.3 Implementace rozesílání v preferovaném kódování

Od tohoto úseku pipeline, tedy po zpracování modulem Decode.py, již počítám s tím, že je zpráva v určitém předem zvoleném kódování (zvolil jsem UTF-8). Pro splnění posledního požadavku skupiny překladatelů bylo potřeba ještě vyřešit převod zprávy na preferované kódování uživatele. Pro tento účel jsem upravil modul Mailmanu, který pracuje na rozesílání zpráv účastníkům.

O odchozí zprávy se stará *OutgoingRunner*, který zpracovává zprávy z adresáře *out*. OutgoingRunner používá podle nastavení pro komunikaci s poštovním serverem handler *SMTPDirect* tak, že volá metodu *process* třídy SMTPDirect. Ve třídě SMTPDirect probíhá několik důležitých operací. V první řadě se volá metoda *chunkify*, ta rozdělí seznam všech adres účastníků na menší seznamy. Vzniklé seznamy se v cyklu prochází a pro každý z nich se zavolá handler Decorate (*Decorate.process*). Třída Decorate provádí překódování zprávy a poslední úpravy před odesláním. Za samotné odeslání je zodpovědná metoda *bulkdeliver* třídy SMTPDirect.

4.3.1 Chunkify

Autoři Mailmanu v rámci jakési optimalizace vytvořili metodu *chunkify*, která třídí adresáty podle domén prvního řádu. Tato funkce byla podle mého názoru implementována kvůli optimalizaci rozesílání resp. optimalizaci překladu doménových jmen, ale protože poštovní server musí každou doménu adres překládat zvlášť, bez ohledu na seřazení adres, metoda nemá žádný praktický přínos. Pro účely nové konference jsem se rozhodl metodu *chunkify* přepracovat tak, aby se příjemci rozdělili do skupin podle zvoleného preferovaného kódování.

Výsledná metoda prochází seznam příjemců a každého z nich zařadí do vhodného seznamu podle kódování, které je k jeho adrese v konferenci přiřazeno. Pro tento účel jsem do slovníku třídy MailList (tato třída reprezentuje konferenci) přidal mapu s adresou jako klíčem a kódováním jako hodnotou. Když v mapě neexistuje klíč s adresou konkrétního uživatele, použije se výchozí hodnota UTF-8, a příjemce se přidá do skupiny s tímto kódováním.

Metoda vrací seznam *chunks*, obsahující seznamy adres účastníků.

Poštovní servery tradičně dovolují během jednoho spojení poslat zprávu na omezený počet příjemců. Přesněji řečeno poštovní server mívá nastavený limit adres v příkazu *RCPT TO* (příkaz, udávající příjemce v SMTP). Typicky bývá toto maximální množství nastaveno na 500 zpráv. Mailman z toho důvodu rozděluje seznamy příjemců v *chunks*, vytvořeném metodou *chunkify*, na skupiny s počtem adres menším než 500 členů. Toto poslední rozdělení již neprobíhá v metodě chunkify, ale tématicky s ní souvisí, proto je uvedené v této sekci.

4.3.2 Vlastní překódování

Modul Decorate jsem shledal jako nejvhodnější místo pro úpravu a přidání funkčnosti převodu kódování, protože je to poslední místo, ve kterém probíhá modifikace zprávy před

odesláním. Zakódování zprávy je provedeno podle již popsaného postupu přes třídu unicode a její metodu encode.

Překódování zprávy by se provádělo pro každého účastníka zvlášť, ale díky vhodnému rozdělení adresátů do skupin podle kódování je počet převodů omezen na počet skupin. Kódování skupiny se určuje podle jejího prvního účastníka, který má nastavenou preferenci.

Tento způsob jsem zvolil proto, že kódování skupiny se nijak v seznamech nepředává. Z toho důvodu musím seznam projít, ale po nalezení první adresy, podle které se dá zjistit kódování, se procházení seznamu ukončuje. V nejlepším případě je počet kontrolovaných adres roven jedné, v nejhorším se testuje celá skupina, což je maximálně 500 adres, než se zvolí výchozí kódování (UTF-8). Nejhorší případ by neměl nastat, když budou mít všichni účastníci kódování nastavené.

Speciální případ nastane při zvolení kódování ASCII. Toto kódování neobsahuje znaky s diakritikou, takže pokud by zpráva tyto znaky obsahovala, došlo by ke značnému poškození textu zprávy. Problém řeši tzv. *transliterace* - odstranění diakritiky. Pro tento proces jsem použil modul *iconvcodec*. Modul bylo potřeba stáhnout[19] a nainstalovat.

K instalaci modulu je potřeba mít nainstalovaný balík *python-dev*,

```
apt-get install python-dev
```

protože obsahuje hlavičkové soubory Pythonu nutné ke kompilaci vestavěného modulu Pythonu. Samotná instalace se provede zkompilováním zdrojových souborů, což zařídí instalační skript *setup.py*.

```
python setup.py build
```

Abych mohl použít *iconvcodec* v Mailmanu, zkopíroval jsem knihovnu do adresáře systému:

```
cp build/lib.linux-i686-2.5/iconv* /var/lib/mailman/Mailman
```

Zjistil jsem, že aby knihovna iconv spolu s transliterací správně fungovala, musí se inicializovat národní prostředí pomocí modulu *locale* a metody *setlocale*. Použití *iconvcodec* pro transliteraci demonstruje následující příklad:

```
>>> import iconvcodec
>>> import locale
>>> locale.setlocale(locale.LC_ALL, "cs_CZ.utf8")

>>> icnv = iconvcodec.iconv
>>> icnv
<module 'iconv' from 'iconv.so'>
>>> cd = icnv.open("ascii//translit", "utf-8")
>>> cd.iconv("ěščřžýáíéúůĚŠČŘŽÝÁÍÉĎÚŮď")
'escrzyaieuuESCRZYAIEDUUd'
```

Do Handleru Decorate jsem připsal podmínku, která rozvětví zakódování podle toho, zda je cílové kódování *ascii*. V této větvi se používá pro zakódování iconvcodec, protože jsem nenalezl způsob, jakým transliteraci provést ve třídě unicode. V opačném případě se běžným způsobem použije encode třídy unicode.

4.3.3 Nastavení uživatelské preference

Aby uživatelé konference mohli nastavit své preferované kódování, bylo potřeba přidat položku do administračního rozhraní systému. Každý účastník má po přihlášení přes webové rozhraní možnost zobrazit seznam účastníků konference a změnit nastavení svého účtu. Tato funkce již v systému Mailman existovala, takže ji stačilo doplnit o nastavení znakové sady.

Webové rozhraní je realizováno pomocí CGI[11] scriptů, skládajících se z wrapperu, napsaného v jazyce C, a příslušných skriptů v Pythonu, které jsou wrappery spouštěny. Úprava webového rozhraní vyžadovala úpravu samotných skriptů v Pythonu, a nebylo potřeba pro ten účel znovu kompilovat wrappery.

Na stránku s nastavením uživatelských účtů jsem přidal seznam dostupných znakových sad, ze kterých si může uživatel zvolit preferované kódování. Volbu jsem přeložil pro tři zprovozněné jazyky - čeština, angličtina a němčina. O vícejazyčnosti systému pojednává další sekce.

4.3.3.1 Úprava skriptů webového rozhraní

Skripty generující webové stránky definují řadu prvků, které se na stránky mohou přidat. Jedním z takových prvků byl *selectbox*, sloužící uživateli pro výběr jazyka.

Ve formě metod jsou prvky dostupné v souboru *HTMLFormatter.py*. Konkrétní metodu *GetLangSelectBox* jsem zkopíroval a upravil na *GetCharsetSelectBox*.

V této nové metodě GetCharsetSelectBox jsem vytvořil seznam znakových sad, ze kterých může uživatel vybrat své preferované kódování.

```
values = [ 'utf-8', 'iso-8859-2', 'cp1250', 'ascii' ]
```

Metoda vygeneruje HTML kód a ten se následně vloží do šablony stránky. V HTML se vygeneruje tento kód:

```
<Select name="pref_charset">
 <option value="utf-8"> utf-8 </option>
 <option value="iso-8859-2" Selected> iso-8859-2 </option>
 <option value="cp1250"> cp1250 </option>
 <option value="ascii"> ascii </option>
</Select>
```

Pro jednoduchost jsem nechal název kódování stejný jako hodnotu, kterou selectbox vrací. Klíčové slovo *Selected* určuje, která hodnota je po načtení stránky vybraná jako výchozí. Vybrána je hodnota, kterou má uživatel vybranou z předchozího uloženého nastavení. Pokud žádné dosud nevybral, výchozí hodnotou je UTF-8.

4.4 Vícejazyčnost konference

Nová e-mailová konference by měla podle zadání podporovat přepínání mezi jazykovými mutacemi. Vícejazyčnost zpráv, které se v konferenci objevují, systém ovlivnit nemůže. Tato

otázka je zcela v režii uživatelů, ale vzhledem k tomu, že se jedná o společnost, která ovládá všechny jazyky, ve kterých se zprávy posílají, není potřeba tento problém řešit. V současnosti se v konferenci objevují zprávy psané v angličtině, ale i v češtině a slovenštině.

Součástí systému Mailman je webové rozhraní, a samotný Mailman je napsán tak, že dokáže zobrazit stránky v různých jazykových mutacích s využitím balíku *gettext*. Navíc e-mailové zprávy, které slouží ke komunikaci mezi systémem a účastníkem (např. ověření adresy), se generují na základě šablony podle aktuální zvolené jazykové mutace. Veškeré šablony a datové soubory, obsahující přeložené texty, jsou v adresářích *messages* a *templates*, umístěných v */var/lib/mailman/*.

4.4.1 Gettext

V adresáři *messages* se nachází datové soubory, se kterými pracuje *gettext*. Gettext je skupina internacionalizačních utilit, dostupná z oficiálního repozitáře Debianu. O projektu gettext je možné se dočíst na oficiálních webových stránkách projektu[18].

Za pomoci nástrojů gettext použije Mailman konkrétní přeložené texty podle zvoleného jazyka. Každý jazyk by měl obsahovat přeloženou množinu zpráv, resp. textových řetězců, které se zobrazují v rozhraní Mailmanu, nebo se používají jako text zpráv, rozesílaných účastníkům. Nejjednodušší způsob podle návodu na stránkách Mailmanu v sekci *často kladené otázky*[12], jak vytvořit nový překlad, je zkopírovat si již existující soubory s přeloženými texty pro některý jazyk a přeložit jednotlivé textové řetězce. V adresáři *messages* se jedná o soubor s příponou *.po* (gettext message catalogue). Po přeložení se soubor s texty musí zpracovat programem *msgfmt*.

```
msgfmt -cv mailman.po
```

Program msgfmt vygeneruje binární soubor (GNU message catalog) a ten se následně používá za běhu Mailmanu, když je potřeba získat překlad konkrétního textu.

4.4.2 Převod kódování jazykových souborů v templates

Z adresáře *templates* se načítá množina textových souborů a šablon webových stránek, kritériem je opět zvolený jazyk. Když jsem se systémem začal pracovat, česká jazyková mutace byla napsána v kódování ISO-8859-2. V rámci sjednocování kódování jsem se rozhodl, že převedu kódování všech souborů české mutace na kódování UTF-8, které je možné použít pro všechny dnes známé jazyky. Ideální by bylo, kdyby se na toto kódování převedly všechny jazykové mutace.

Pro překódování jsem použil běžně dostupný linuxový nástroj *iconv*, který slouží pro práci s různými znakovými sadami nejen z příkazové řádky. Program iconv jednoduše překóduje text z jednoho kódování do druhého.

V adresáři *templates* jsou jednotlivé jazyky roztříděny do podadresářů podle jednoznačného jména jazyka. Při převodu znakové sady pro český překlad jsem v *bash*i vytvořil jednoduchý cyklus, který s použitím programu iconv práci provedl.

```
for i~in *; do\
 iconv -f iso-8859-2 -t utf-8 $i > $i.conv && mv $i.conv $i;\
done
```

Cyklus byl spuštěn z adresáře, ve kterém se šablony nachází. Důležité bylo převedená data ukládat do nového souboru a až potom nahradit původní soubor. Přesměrování do soboru se stejným názvem by totiž znamelo ztrátu dat, protože bash nejprve "vytvoří" cílový soubor (tj. přepíše ho) a pak začne číst ze zdrojového (který už je ale v tu chvíli prázdný).

4.4.3 Přidání jazyka do konference

Seznam jazyků, které se zobrazují v nabídce konference, se načítá z proměnné *available_languages* třídy MailList. To znamená, že když máme přeložené texty z *messages* a *templates* a tyto texty uložené v podadresářích se jménem jazyka, můžeme jazykovou mutaci přidat na seznam dostupných jazyků konference modifikací proměnné *available_languages*.

Přidání jsem provedl jednoduchým příkazem v konzoli Pythonu. Po importování potřebných cest Mailmanu a třídy MailList se otevře konference zadaná jménem (v tomto případě "czechlist"). Do seznamu *available_languages* můžeme přidat položku metodou *append* se jménem jazykové mutace jako parametrem. Provedení modifikace seznamu demonstruje následující příklad, který zařadí německý jazyk mezi dostupné jazyky nové konference.

```
>>> import paths
>>> from Mailman import mm_cfg
>>> from Mailman import MailList
>>> mlist = MailList.MailList( 'czechlist' )
#konstruktor zároveň i~zamkne konferenci
>>> mlist.available_languages.append( 'de' )
>>> mlist.Save()
>>> mlist.Unlock()
```

Aby se dal jednoduše importovat modul *paths* jako je to v příkladu, je dobré spustit interpret pythonu z adresáře */var/lib/mailman/bin*.

Kapitola 5

Testování

Testování systému probíhalo po celou dobu realizace konference, a to ruční kontrolou správnosti přijatých zpráv. V první fázi, když se podařilo zprovoznit systém Mailman, jsem zkoušel odesílání a doručování pokusných zpráv mezi uzavřenou skupinou e-mailových adres bez jakékoliv interakce s existující překladatelskou konferencí.

5.1 Testovací skupina na Yahoo! Groups

První problém, který bylo nutné vyřešit, bylo napojení nově vzniklého systému pro konferenci na službu Yahoo! Groups.

Pro tento účel jsem zaregistroval testovací skupinu na Yahoo! Groups[23] pod jménem *ba-maillist-czechlist*, abych mohl simulovat provoz mezi živou verzí e-mailové konference skupiny překladatelů a novým serverem, který by měl začít fungovat paralelně s konferencí na Yahoo!. Doručování testovacích zpráv se tím rozšířilo mezi dvě konference.

Před prvním spuštěním se objevily obavy ze vzniku jakési smyčky při přeposílání zpráv. Tato smyčka by mohla vzniknout, pokud by obě dvě strany (server Yahoo! i server se systémem Mailman) posílali kopii odeslané zprávy i tomu, kdo zprávu odeslal. Kopie zprávy vypadá jako by ji odeslal server (obsahuje přepsané hlavičky). Kdybych tedy propojil dvě konference mezi sebou bez ohledu na tento problém, hrozilo by riziko, že jakákoliv zpráva zaslaná do jedné konference vyvolá odeslání té samé zprávy do druhé konference. Až potud je chování v pořádku, ale když bude cílová konference brát odesílatele (adresu konference) jako běžného účastníka konference, pošle mu jeho zprávu nazpět. Situace se bude opakovat, tímto způsobem by vznikla nekonečná smyčka. Zjistil jsem, že toto je standardní chování e-mailových konferencí, ale musel jsem najít způsob, jak toho chování obejít.

Řešení tohoto problému jsem objevil přímo v konfiguraci účastníka v obou konferencích. V systému Mailman i při registraci na Yahoo! Groups existuje možnost zakázat posílání kopie vlastních zpráv. Na základě tohoto opatření nemůže dojít k výše uvedené smyčce, a konference jsou bez potíží propojeny. Každá zpráva by se měla dostat ke každému účastníkovi obou konferencí. Nabyté vědomosti během pokusů s ba-maillist-czechlist jsem použil při napojování nové konference czechlist.org na stávající konferenci na Yahoo!.

Tato fáze zahrnovala testování doručování zpráv a ověření, že nedojde během provozu ke vzniku smyčky.

5.2 Testování za provozu

Po otestování spolupráce se zkušební skupinou na Yahoo! Groups došlo k propojení existující fungující konference překladatelů a nové konference czechlist.org. Od té chvíle jsem testovací příspěvky posílané do konference omezil na minimum a testování probíhalo jen na základě reálných příspěvků překladatelů.

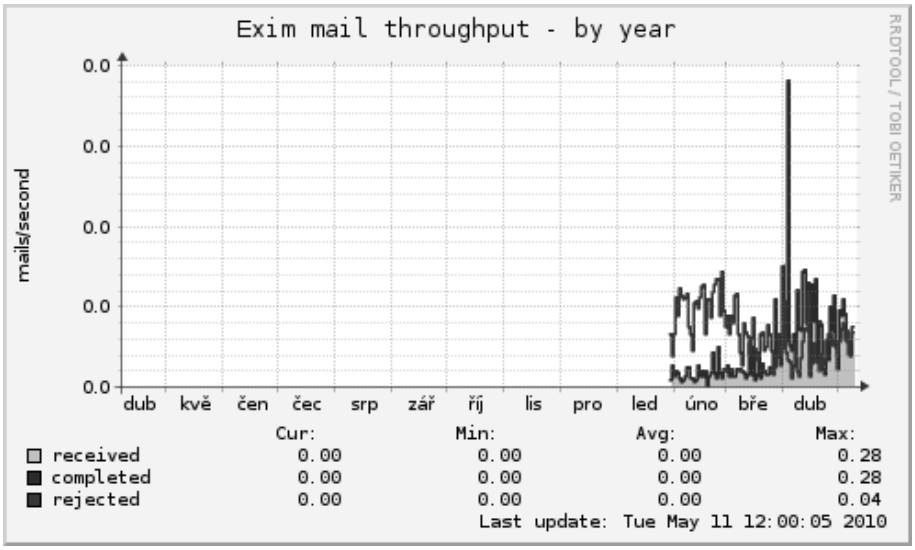
Další součástí testování byla registrace nezávislého e-mailového účtu, na který by byly odesílány všechny zprávy překladatelů po propojení obou konferencí. Na výběr jsem měl spoustu poskytovatelů e-mailových služeb zdarma. Zaregistroval jsem účet *joeczechlist@seznam.cz*, aby bylo možné sledovat úspěch při rozpoznávání kódování zpráv.

5.3 Sledování činnosti

Monitorování systému zajišťuje systém Munin[21], dostupný pod licencí GNU GPL. Munin je systém s architekturou klient-server, napsaný v jazyce Perl. Jeho výstupem jsou grafy, znázorňující naměřené hodnoty. Všechny grafy jsou zhlédnutelné na webových stránkách serveru Munin ve formě obrázků PNG.

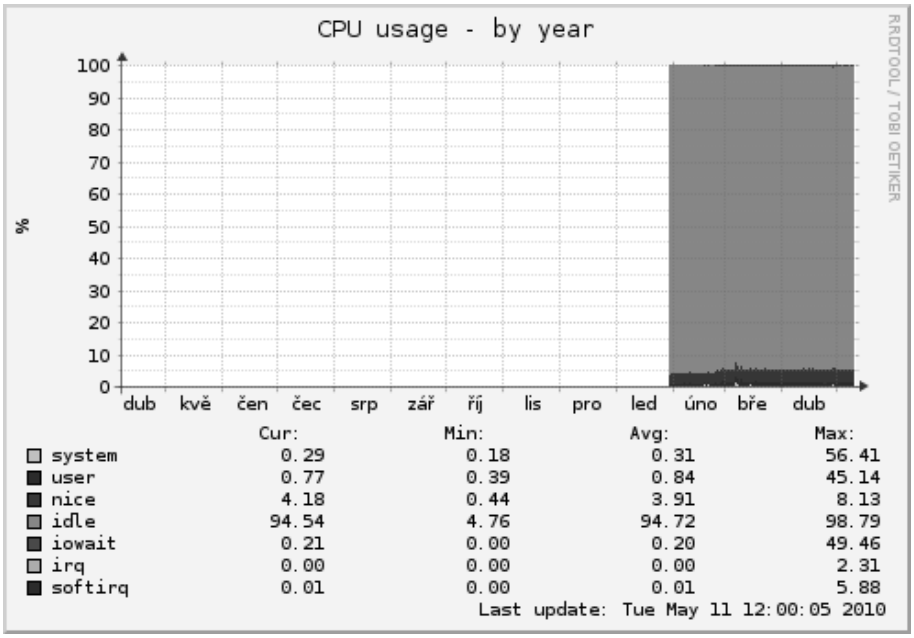
Jako ukázku grafů vytvořených systémem Munin jsem vybral statistiku poštovního serveru a využití hlavního procesoru serveru, na kterém byla konference realizována.

Z grafu poštovního serveru Exim je vidět několikanásobný nárůst příchozích a odchozích zpráv po spuštění nové konference. Toto zvýšení bylo očekávané a není nijak závažné, protože počet zpráv, které poštovní server před nasazením Mailmanu doručoval, byl minimální.



Obrázek 5.1: Graf zatížení poštovního serveru

Graf využití procesoru nám říká, že systém Mailman procesor nijak nevytěžuje, využití se drží na hranici 10%. Systém má podle těchto faktů výkonostní rezervu a měl by provoz konference bez problémů zvládat.



Obrázek 5.2: Graf využití CPU

Pro doplnění informací je nutné dodat časové souvislosti. Nasazení systému Munin proběhlo na konci měsíce ledna. Od té doby probíhalo lokální testování e-mailové konference, tedy testování pokusnými zprávami. Někdy na přelomu března a dubna došlo k propojení Mailmanu a existující překladatelské konference na Yahoo! Groups.

Kapitola 6

Závěr

6.1 Zhodnocení

Během mé práce se podařilo zprovoznit a nastavit fungující systém pro e-mailovou konferenci, který může skupina překladatelů využívat jako alternativu k doposud používaným službám. Nově nasazený a mnou upravený systém řeší problémy, se kterými se překladatelé při práci potýkali.

Zprávy, přijaté do nové e-mailové konference, se nyní daří úspěšně zobrazit po rozpoznání jejich kódování. Do konference na Yahoo! se přeposílají z nové konference zprávy ve znakové sadě, která je správně zobrazitelná v tamějším webovém archivu zpráv. Problém by mohl nastat u zpráv, které obsahují citované části jiných zpráv. U takových zpráv se občas vinou některého poštovního klienta stává, že jednotlivé části nejsou ve stejném kódování. V další sekci "Pokračování práce" jsem uvažoval o možných řešení tohoto problému. Po přechodu všech uživatelů do nové konference tato situace nenastane, protože po nastavení preferovaného kódování budou uživatelům přicházet příspěvky jen v jednom konkrétním kódování a tudíž ani odpovědi nebudou obsahovat citace, jejichž kódování by se lišilo. Proto je problém, který se však objevuje jen občasně, pouze přechodný a po úplném nasazení systému vymizí úplně.

Rychlost doručení zpráv se pohybovala během testování nového systému v řádu desítek sekund až minuty, což je v porovnání se službou Yahoo! Groups mnohonásobně rychlejší - některé zprávy se zpozdí v průměru až o desítky minut. Z tohoto pohledu je projekt velmi úspěšný.

Posledním důležitým požadavkem byla vícejazyčnost konference. Vzhledem k vhodnému výběru systému, na kterém jsem se rozhodl konferenci zrealizovat, byla podpora jazykových mutací splněna od začátku. Ve své práci jsem popsal způsob, jakým přidat novou jazykovou mutaci do systému Mailman a jak následně rozšířit množinu dostupných jazykových verzí konference.

6.2 Pokračování práce

Přesto, že citované části zpráv do budoucna nebudou působit výše zmíněné komplikace, pokud budou všichni účastníci používat nový systém, uvažoval jsem o možném řešení. Zjistil

jsem, že některé zprávy, které přijdou na začátku správně, se následně v průběhu komunikace uživatelů objeví se špatnými znaky. Jedná se o případy, kdy se předchozí text zprávy nebo jeho část zkopíruje jako citát a uživatel připíše do zprávy svůj text. Nová část zprávy bývá zobrazena v pořádku a citované části obsahují chyby v zobrazení znaků. Možné řešení spočívá v práci s citovanými částmi. Výhodou se může v této situaci zdát úzus označování citovaných částí zpráv. Řádky, které jsou citované z nějaké předchozí zprávy, většinou začínají znakem '>'.

Na základě tohoto faktu by bylo možné rozdělit každou zprávu na dílčí části podle toho, jedná-li se o citaci nebo novou část textu. Případně je možné i určit, jaké "úrovně citování" je konkrétní část (myšleno z pohledu jakéhosi stromu, který bychom si mohli z textu vytvořit za předpokladu, že každý uzel by byl tvořen jednou citovanou odpovědí, a hrany propojující uzly by vyznačovaly vztah mezi dvěma po sobě jdoucími odpověďmi).

Na jednotlivé části takto rozdělené zprávy by poté bylo možno aplikovat rozpoznávací algoritmus, který by se snažil rozpoznat znakovou sadu. Tímto způsobem by se dal vyřešit problém špatného zobrazení zpráv, které obsahují texty více než jednoho kódování. Řešení by ale bohužel nebylo univerzální a vyžadovalo by striktní dodržování významu znaku '>' na začátku řádky.

Dále se nabízí doplnění algoritmu rozpoznávání kódování v případě, že použitý program *enca* nedokáže rozhodnout. V tomto případě by se mohlo jako doplnění testů rozkódování použít nasazení libovolného programu na kontrolu pravopisu, tzv. *spellchecker*u. V linuxovém prostředí se pro kontrolu pravopisu používá například GNU Aspell. Idea je taková, že po pokusu rozkódování textu zkoušeným kódováním se pustí spellchecker na výsledná data, a tím se získá kritérium pro porovnávání testů znakových sad, u kterých nejde jednoznačně rozhodnout o správnosti. Nutnou podmínkou použití této metody by ale byla nutná znalost jazyka textu.

Literatura

[1] D. M. Beazley. Advanced Python Programming, 2000.
O'Reilly Open Source Conference,
prezentace dostupná na `http://www.xminc.com/mt/archives/pythontut2.html`,
stav z 11.5.2010.

[2] M. Dethmers. phplist.com : Homepage, 1999.
`http://www.phplist.com/`, stav ze 17. 4. 2010.

[3] E. Harris. The Next Step in the Spam Control War: Greylisting. 2003.
`http://www.greylisting.org/articles/whitepaper.shtml` revisited 21.8.2003
stav z 15.5.2010.

[4] S. Kemp. Running a mailing list with Mailman, 2005.
`http://www.debian-administration.org/articles/108`, stav z 18. 4. 2010.

[5] N. Metheringham. exim internet mailer, 2007.
`http://www.exim.org/`, stav z 19. 4. 2010.

[6] N. Freed, Innosoft, N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One, 1996.
`http://www.ietf.org/rfc/rfc2045.txt`, stav z 1. 5. 2010.

[7] J. B. Postel. Simple Mail Transfer Protocol, 1982.
`http://www.ietf.org/rfc/rfc821.txt/`, stav ze 3. 4. 2010.

[8] Python Software Foundation. pdb — The Python Debugger.
`http://docs.python.org/library/pdb.html`, stav z 18. 4. 2010.

[9] Python Software Foundation. Python Programming Language, 1990.
`http://www.python.org/`, stav ze 13. 4. 2010.

[10] Python Software Foundation. Python v2.6.5 documentation, 1990.
`http://doc.python.org/`, stav ze 13. 4. 2010.

[11] L. Quin. CGI - Common Gateway Interface, 2009.
`http://www.w3.org/CGI/`, stav ze 17. 4. 2010.

[12] C. Siddall. i18nhowto - development - confluence.
`http://wiki.list.org/display/DEV/i18nhowto`, stav z 24. 4. 2010.

[13] J. Simoni. Dada Mail is a Contemporary Mailing List Manager, 1999.
http://www.dadamailproject.com/, stav ze 17. 4. 2010.

[14] Software in the Public Interest, Inc. Debian – univerzální operační systém, 2010.
http://www.debian.org/, stav ze 17. 4. 2010.

[15] The Apache Software Foundation. Project Details for Apache HTTP Server, 1996.
http://projects.apache.org/projects/http_server.html, stav z 19. 4. 2010.

[16] DomainKeys Identified Mail (DKIM).
http://www.dkim.org/, stav z 20. 4. 2010.

[17] Howto: Using exim4 and mailman together, 2007.
http://www.exim.org/howto/mailman21.html#roconf, stav z 19. 4. 2010.

[18] gettext - GNU Project - Free Software Foundation (FSF).
http://www.gnu.org/software/gettext/, stav z 24. 4. 2010.

[19] Modul iconvcodec.
http://pypi.python.org/packages/source/i/iconv/iconv-1.0.tar.gz,
stav z 10. 5. 2010.

[20] Mailman, the GNU Mailing List Manager.
http://list.org/, stav ze 3. 4. 2010.

[21] Munin - Trac.
http://munin-monitoring.org/, stav z 11. 5. 2010.

[22] Spam - Wikipedie otevřená encyklopedie.
http://cs.wikipedia.org/wiki/Spam, stav ze 3. 4. 2010.

[23] Yahoo! Groups.
http://groups.yahoo.com/, stav ze 3. 4. 2010.

[24] M. Čihař. Enca - freshmeat.net.
http://freshmeat.net/projects/enca/, stav z 23. 4. 2010.

Dodatek A

Použitá konfigurace webového serveru Apache

```
<VirtualHost *:80>
        ServerAdmin webmaster@kassoft.cz
        ServerName www.czechlist.org
        DocumentRoot /var/www/czechlist
        AddDefaultCharset UTF-8
        ScriptAlias /mailman/ /usr/lib/cgi-bin/mailman/listinfo
        ScriptAlias /cgi-bin/mailman/ /usr/lib/cgi-bin/mailman/
        ScriptAlias /cgi-bin/ /home/www/czechlist/cgi-bin/
        <Location /cgi-bin>
                Options +ExecCGI
        </Location>
        <Directory /usr/lib/cgi-bin/mailman/>
            AllowOverride None
            Options ExecCGI
            Order allow,deny
            Allow from all
        </Directory>
        Alias /pipermail/ /var/lib/mailman/archives/public/
        <Directory /var/lib/mailman/archives/public>
            Options Indexes MultiViews FollowSymLinks
            AllowOverride None
            Order allow,deny
            Allow from all
        </Directory>
        Alias /mmimages/ /usr/share/doc/mailman/images/
        <Directory /usr/share/doc/mailman/images>
            Options Indexes MultiViews
            AllowOverride None
            Order allow,deny
            Allow from all
```

```
        </Directory>
</VirtualHost>
```

Jednotlivé sekce nastavují cesty k prostředkům Mailmanu jako jsou skripty, obrázky atd.

Dodatek B

Instalační příručka

Za účelem zprovoznění serveru e-mailové konference je nutné nainstalovat systém Mailman včetně dalších potřebných programů. Doporučený operační systém, na kterém byl pro účely této práce systém zprovozněn, je:

```
Debian GNU/Linux 5.0.4 (Lenny), verze jádra 2.6.26-2-686
```

Mezi potřebné balíky patří:

1. apache2
2. exim4
3. mailman
4. python (>= 2.3)
5. python-dev

Po instalaci a nastavení konference stačí přepsat skripty v adresáři */usr/lib/mailman/Mailman* upravenou verzí. Instalace a konfigurace systému je přiblížena v textu této práce konkrétně v kapitole Realizace.

Dodatek C

Obsah přiloženého CD

```
.
|-- usr
|   `-- lib
|       `-- mailman
|           |-- bin
|           |-- cron
|           |-- mail
|           |-- Mailman
|           |   |-- Archiver
|           |   |-- Bouncers
|           |   |-- Cgi
|           |   |-- Commands
|           |   |-- Gui
|           |   |-- Handlers
|           |   |-- Logging
|           |   |-- MTA
|           |   `-- Queue
|           |-- pythonlib
|           |   `-- email
|           `-- scripts
`-- var
    `-- lib
        `-- mailman
            |-- archives
            |   |-- private
            |   |   |-- czechlist
            |   |   |   |-- database
            |   |   `-- czechlist.mbox
            |   `-- public
            |-- data
            |-- lists
            |   `-- czechlist
```

```
|-- messages
|   |-- cs
|   |-- de
|-- qfiles
|   |-- archive
|   |-- bad
|   |-- bounces
|   |-- commands
|   |-- in
|   |-- news
|   |-- out
|   |-- retry
|   |-- shunt
|   `-- virgin
|-- spam
`-- tests
    |-- bounces
    `-- msgs
```

Obsah CD vznikl zkopírováním souborů ze dvou hlavních adresářů systému Mailman. Jedná se o */var/lib/mailman* a */usr/lib/mailman*.

Tento seznam obsahuje pouze adresáře a pro přehlednost jsem ve výpisu vynechal adresáře jazykových mutací. Všechny skripty důležité pro činnost systému se nachází v adresáři *usr/lib/mailman*. Po instalaci balíčkovacím systémem jsou všechny soubory Mailmanu dostupné z adresáře */var/lib/mailman* pomocí symbolických odkazů, ty jsem na CD nekopíroval.