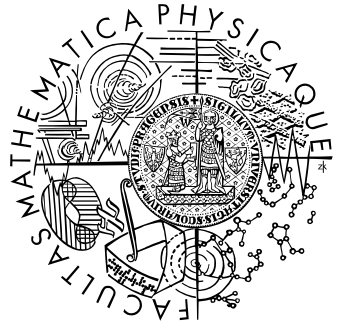
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Štefan Alakša

Komprese e-mailu

Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Jan Lánský

Studijní program: Informatika, Správa počítačových systémů

2009

Na tomto místě bych chtěl poděkovat zejména svému vedoucímu bakalářské práce Mgr. Janu Lánskému za rady a náměty pro zdokonalení práce a za pomoc při odstraňování chyb. Rovněž bych mu chtěl poděkovat za velmi přínosné publikace týkající se komprese textových dat.

Dále bych chtěl poděkovat přátelům, kteří poskytli souhlas s použitím svých e-mailových zpráv v bakalářské práci pro účely vývoje, ladění a srovnání účinnosti kompresních programů a potažmo s jejich zveřejněním.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 6.8.2009 Štefan Alakša

Obsah

Název práce: Komprese e-mailu
Autor: Štefan Alakša
e-mail autora: stefanovi@seznam.cz
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí bakalářské práce: Mgr. Jan Lánský
e-mail vedoucího: jan.lansky@mff.cuni.cz

Abstrakt: E-mail je velmi specifická datová struktura upravená příslušnými normami RFC. Běžná komprese pro textová data použitá na emaily nezohledňuje strukturu e-mailu, nejvíce je patrná nízká účinnost běžných metod u malých souborů, v nichž velikost hlavičky často převažuje nad délkou vlastního těla zprávy, nebo naopak u e-mailů s velkými zakódovanými přílohami. Tato bakalářská práce se pokouší navrhnout a implementovat vhodný algoritmus na kompresi tohoto typu dat. Rozdělením e-mailu na hlavičky a vlastní obsah, samostatnou kompresí hlaviček vlastním algoritmem založeném na slovníkové kompresi a kompresí těla a příloh dle MIME typu jednotlivých částí vhodnými existujícími programy a algoritmy se pokouší zvýšit účinnost komprese tohoto typu dat.

Klíčová slova: komprese textu, dělení na slova, struktura e-mailu, algoritmus LZW, Huffmanovo kódování

Title: Compression of e-mails
Autor: Štefan Alakša
Author's e-mail address: stefanovi@seznam.cz
Department: Department of software engineering
Supervisor: Mgr. Jan Lánský
Supervisor's e-mail address: jan.lansky@mff.cuni.cz

Abstract: E-mail is a very specific data structure described in appropriate RFC standards. Common text-based compression applied to e-mails does not regard e-mail structure. Low compression ratio of common methods is noticeable on small files where header size often exceeds message body length or on e-mails with big encoded attachments. This bachelor thesis tries to suggest and implement a suitable algorithm for this type of data. By separating an e-mail message into headers and content, compressing headers with own dictionary-based algorithm, body and attachments with suitable existing programs or algorithms according to MIME type of the parts, separately, it tries to increase efficiency of compression of this type of data.

Keywords: text compression, word-based model, e-mail structure, LZW algorithm, Huffman coding

1. Úvod

Znalost struktury komprimované zprávy je užitečná pro návrh účinného kompresního algoritmu. Při kompresi e-mailu je struktura textová, definovaná normami RFC[1] 822, 2822, 5322, 5335 (struktura e-mailu jako takového) a RFC 2045-2049 (rozšíření MIME).

Tato struktura je na jednu stranu přesně vymezená. Přesto je natolik složitá a variabilní, že mnoho emailových klientů a programů normu nedodržuje. Kompresní algoritmus musí být schopný přijmout jakákoli data, tedy i neplatný e-mail, za cenu zhoršení kompresního poměru nebo odmítnutí zpracování v případě, že komprimovaná zpráva by byla ve skutečnosti větší, než původní data.

Aby zpráva byla e-mailem, musí splňovat základní strukturu. Na začátku e-mailu je hlavička. Je to několik řádků 7bitového textu následovaných prázdným řádkem. Následuje buď jediné tělo zprávy, nebo, je-li v hlavičce definováno tělo jako MIME typ message nebo multipart, je tělo vnořenou zprávou opět obsahující vlastní hlavičku a tělo. Vnořování není omezeno, e-mail tak má strukturu stromu, přičemž hodnoty (tělo) se nacházejí pouze a ve všech listech.

Na začátku máme tyto předpoklady a očekávání:

P1: Komprese e-mailů bude zohledňovat různé typy dat, bude jim rozumět z hlediska informace, kterou reprezentují. Tedy na jedné úrovni bude umět rozeznat hlavičku, tělo, nekódovanou i kódovanou přílohu, na nižší úrovni u hlaviček pak jednotlivé typy dat: slova, čísla, znaky, složené typy (datum, čas, IP adresa, typ kódování), a použití různé reprezentace různých typů dat bude mít za následek zlepšení kompresního poměru.

P2: Slova povinná nebo velmi častá, vyskytující se v hlavičce platného e-mailu, nemá smysl přidávat do dat, informace o znění slova tak ponese algoritmus sám, jeho součástí bude struktura používající zabudovaný slovník.

P3: Účinnost komprese by se podle P2 měla nejvíce projevit na souborech, kde je velikost hlaviček alespoň polovinou velikosti celého e-mailu.

P4: Rozdíl v účinnosti komprese by se měl projevit i u e-mailů s velkými kódovanými přílohami (odstraněním redundantního kódování dojde ke zmenšení velikosti dat vstupujících do komprese).

[1] Request For Comments: http://www.ietf.org/rfc.html

Komprese e-mailu musí vyřešit tyto problémy:

- zda je lepší pro každou hlavičku vyrábět zvláštní strukturu, tu naplňovat a následně úspornou metodou uložit, nebo je lepší brát hlavičky jako text bez přesné normy a ukládat úsporně až jednotlivé typy dat, a to z ohledu efektivity časové i kompresní,
- jakým způsobem vybrat vhodný algoritmus pro tělo nebo už dekódovanou přílohu,
- jak reprezentovat strom hlaviček,
- v případě slovníkové metody, jak reprezentovat vlastní a zabudovaný slovník a zda je nesloučit do jednoho.

V následující kapitole se zkoumá strukturu e-mailu, nastíní možné přístupy jejího zpracování. Ve třetí kapitole jsou vybrána rozmanitá data k analýze a testování, čtvrtá kapitola se pouští do rozdělení e-mailu na logické celky a struktury, v páté kapitole je navržen algoritmus pro kompresi hlaviček. V šesté je měřena účinnost navržených postupů na dvou typických skupinách dat ze třetí kapitoly, v sedmé a osmé se nachází dokumentace k programům, které byly pro tuto úlohu vytvořeny. Na závěr se zhodnotí výsledky a navrhnou způsoby, jakým směrem pokračovat ve vývoji.

2. Struktura emailu, vhodné kompresní algoritmy

Jak bylo zmíněno v úvodu, e-mail je dokument, který obsahuje hlavičky v přesně popsané formě[2] a vlastní tělo, oddělené od hlaviček prvním prázdným řádkem, přičemž tělo může být vnořeným e-mailem nebo řadou vnořených e-mailů. Základní strukturou je tedy posloupnost neprázdných řádků, prázdný řádek a zbytek souboru. Zbytek souboru může být tělo zprávy nebo posloupnost jednoho nebo několika vnořených e-mailů. Tuto strukturu budeme u e-mailu předpokládat.

2.1. Hlavičky

Úvodní hlavička e-mailu se skládá z řádků, každý z nich z klíčového slova (zde bude označováno jako vlastnost v hlavičce), dvojtečky, bílých znaků a hodnoty. Případně může následovat středník a za ním další text. Norma připouští také komentáře, ty se v hlavičce téměř nepoužívají. Řádky odsazené bílým místem mohou mít jiný formát. Pro celou hlavičku platí, že používá pouze tisknutelné nebo bílé znaky s kódem ASCII 9 (tab), 10 (LF), 13 (CR), 32 (mezera) až 126 (~). Konce řádků jsou vždy oba znaky CRLF v uvedeném pořadí.

Délka řádku v e-mailu je doporučená 80, maximální 1000 znaků včetně znaků konce řádku. To dovoluje při zpracování po řádcích použít řetězec omezené délky.

V informacích obsažených v hlavičce se nachází i informace o odesílateli, příjemci, cestě po síti Internet a mnoho různých dat doplněných e-mailovými klienty, servery a antivirovými programy, většinu z nich však k použití účinnější komprese nepotřebujeme. Důležité jsou informace o typu a kódování těla. V hlavičce by se měla nejvýše po jednom vyskytovat klíčová slova standardu MIME[3], tedy slova MIME-Version, Content-Type, Content-Transfer-Encoding, Content-ID, Content-Description a kódovaný tvar textu.

Pro analýzu a dekódování obsahu e-mailu jsou důležité dvě vlastnosti, a to Content-Type, ze které se dozvíme, zda je tělo jednoduché nebo jde o vnořený e-mail (typy multipart/*, message/rfc822), a Content-Transfer-Encoding, ze které se dozvíme způsob kódování. Možné jsou 7bit, 8bit, binary, base64, quoted-printable.

[2] P. Resnick (2001): RFC 2822
[3] N. Borenstein, N. Freed (1996): RFC 2045

Schéma základní struktury e-mailu je tedy následující:

```
e-mail := jeden_díl | více_dílů | vnořený_e-mail

jeden_díl := hlavička_1 prázdný_řádek tělo

více_dílů := hlavička_n prázdný_řádek prázdné_tělo

        [ --oddělovač e-mail ]+ --oddělovač--

vnořený_e-mail := hlavička_11 prázdný_řádek e-mail
```

`hlavička_1` je hlavička, která neoznačuje vícedílné tělo nebo vnořenou zprávu. Je určena absencí řádku začínajícího `Content-Type`, nebo má tato vlastnost hodnotu jinou než `multipart` nebo `message`. V takovém případě se hledá ještě informace o kódování těla ve vlastnosti `Content-Transfer-Encoding`.

`hlavička_11` je hlavička, které označuje jeden vnořený e-mail. Pro náš algoritmus to především znamená, že nebude za prázdným řádkem následovat tělo, protože bude následovat ještě další hlavička. Vlastnost `Content-Type` má v tomto případě hodnotu `message/rfc822`.

`hlavička_n` je hlavička pro vícedílné tělo. Poznáme ji podle hodnoty začínající `multipart` u vlastnosti `Content-Type` a přítomnosti dalšího parametru za středníkem nebo odsazeného na dalším řádku, oddělovače `boundary`.

`prázdné_tělo` je prázdný řádek nebo několik slov s informací, že se jedná o vícedílnou zprávu ve formátu MIME. Tento řádek se při běžném zobrazení e-mailu objevuje pouze v programech, které vícedílné zprávy nepodporují.

`tělo` je samotný obsah sdělení e-mailu, může se jednat o text, HTML dokument nebo soubor – přílohu. Tato část zprávy může být kódována.

Použité operátory v schématu: `|` značí jednu z variant (nebo), `[]+` značí libovolný nenulový počet opakování schématu v závorkách, `--` jsou dvě pomlčky před, případně i za oddělovačem.

Jsou dvě možnosti, jak hlavičky komprimovat. Buď se vyrobí struktura, která bude obsahovat povinné a volitelné vlastnosti uvedené v hlavičce společně s jejich hodnotami. Ta se pak zlinearizuje a uloží binárně. Druhým přístupem je komprimovat hlavičky jednoduše jako obyčejný text, ale využít přitom znalosti struktury k určení a komprimaci různých datových typů. Je to způsob, který kombinuje kompresi textu

s kompresí pomocí binární datové struktury nikoli pro celou hlavičku, ale pro nejčastěji se vyskytující datové typy.

Vzhledem k tomu, že formát e-mailu se může časem měnit, mohou přibýt nové vlastnosti zapsané v hlavičce, parsování hlavičky jako celku spolu s kontrolami korektnosti dat je časově náročné a mnoho e-mailových klientů a serverů nedodržuje normy, považuje autor za lepší druhý způsob, který komprimuje hlavičku jako proud textu, ve kterém však rozeznává jednotlivé datové typy na úrovni řádku a slov. Výhodou tohoto způsobu je bezeztrátovost komprese na pořadí řádků a bílé znaky, rychlost zpracování oběma směry, přehlednost algoritmu, schopnost srovnatelně komprimovat e-maily i po změně normy, akceptace většiny nekorektních e-mailů s výjimkou vnořeného kódování, které norma výslovně zakazuje[4], a nesmyslných vstupů (příliš dlouhé řádky, binární soubory, není patrná stromová struktura hlaviček a těl). Nevýhodou pak je horší kompresní poměr než při využití jedné sofistikované datové struktury pro celou hlavičku. Komprese v detailech není úplně bezeztrátová – při rozkódování a novém zakódování přílohy dochází např. k jinému dělení řádků, zakódování znaků, které původně kódovány nebyly atp. Obsah zprávy a údaje v hlavičkách však zůstávají nezměněny, komprese je tedy obsahově bezeztrátová.

2.2. Textové tělo zprávy

Není-li v hlavičce e-mailu informace standardu MIME, veškerý další obsah za prázdným řádkem oddělujícím hlavičku od těla je považován za jeden nekódovaný soubor. Takové e-maily se však dnes již téměř nevyskytují. Proto je nutné s MIME počítat vždy.

Je-li tedy typ MIME v hlavičce definován, zpracování těla e-mailu se řídí tím, zda jde o typ vícedílný (multipart), vnořený e-mail (message/rfc822) nebo jednodílný, již obsahující samotná data (ostatní označení typu).

V případě, že se jedná o vícedílný typ nebo vnořený e-mail, zpracování se rekurzivně volá od začátku a běží, dokud není přítomen ukončující oddělovač (boundary, definován u vlastnosti Content-Type). Zde je nutno připomenout, že se tělo předem nedekóduje a soubor, který obsahuje kódování Quoted-Printable nebo Base64 na vícedílné tělo, nebude proto účinně zkomprimován.

[4] [3], kapitola 6.4

E-mailoví klienti tělo e-mailu často ukládají ve dvou formátech: text e-mailu v plném formátování (HTML), text bez formátování a tagů, přílohy. První dva se kódují pomocí 8bit nebo Base64, zatímco přílohy, pro zachování binární podoby, pomocí Quoted-Printable.

2.3. Přílohy

V případě jednodílného těla je jeho obsahem libovolný soubor, který je před kompresí dekódován, je-li přítomno kódování Base64 nebo Quoted-Printable.

7bit, 8bit a binary[5] jsou kódování, která není třeba zvláštním způsobem dekódovat. Znamenají, že data jsou všechna 7bitový text (7bit v 8 bitech, nejvyšší bit je vždy 0), 8bitový text (8bit) nebo plně binární (binary). Konec kódovaného obsahu lze zjistit z přítomnosti oddělovače (boundary) nebo hodnoty Content-Length, která označuje délku již zakódovaného obsahu v bytech.

Base64 je binárně bezpečný překlad dat z 8bitové formy do 6bitové přerozdělením z osmic na šestice, ty jsou ještě pomocí převodní tabulky přeloženy na 64 možných tisknutelných znaků anglické abecedy, které jsou nakonec uloženy textově. Zakódováním tak data přiberou 33% své původní velikosti. Toto kódování se vyskytuje nejčastěji u netextových příloh, nebo příloh obecně, pokud je nutné při kódování zachovat přesnou binární reprezentaci souboru a nelze použít kódování binary.

Quoted-Printable je textové kódování, ve kterém všechny netisknutelné znaky, rovnítko a znaky s ASCII kódem větším než 127 jsou nahrazeny třemi znaky: rovnítkem a hexadecimálním kódem nahrazovaného znaku. Konce řádku se nekódují, bílé znaky těsně před koncem řádku se kódovat musí. Výhodou tohoto kódování je dobrá čitelnost, zvláště u anglických textů, nízký nárůst na velikosti a rychlé přímočaré dekódování. Nevýhodou je vysoká míra redundance u kódovaných znaků, nevhodným použitím tohoto kódování na binární soubory lze docílit velikost jejich kódované verze až na 300% původní velikosti.

Odstraněním kódování lze analyzovat původní soubor a zvolit algoritmus, který se pro něj hodí nejlépe. Možností, jakým způsobem vybrat vhodný algoritmus, je více, lze se řídit formátem souboru (z MIME typu nebo analýzou jeho prvního kB), případně jich použít více a nejlepší výsledek zahrnout do souboru komprimovaného e-mailu.

[5] Popis všech kódování je součástí [3] v kapitolách 2.7-2.9, 6.7-6.8

3. Testovací množiny e-mailů

Pro účely vývoje a testování kompresních metod jsou potřebná vzorová data. Testovací e-maily by měly odrážet běžné použití struktury e-mailu, ale nesmí chybět i méně obvyklé případy.

3.1. Enron Corpus

V roce 2001 zbankrotovala americká energetická společnost Enron. Během vyšetřování spojeného s údajným falšováním účetnictví proběhla analýza cca půl druhého milionu e-mailů. Velké množství z nich se dostalo na veřejnost a jsou pod názvem Enron Corpus[6] díky své rozsáhlosti žádaným zdrojem pro nejrůznější vědecké studie.

Celý soubor zabírá půl gigabytu v komprimované (gzip) a přibližně o gigabyte více v nekomprimované (XML) podobě. Z tohoto souboru bylo vybráno 136 e-mailů, všechny bez příloh a zbaveny těla, pro statistickou analýzu hlaviček (množina E_{136}) a 10 e-mailů, většinou s přílohami, pro analýzu obsahu a testování účinnosti (množina E_{10}).

Protože jsou e-maily uložené v XML jako přeposílaný 7bitový text, byla množina E10 upravena tak, aby reprezentovala původní e-maily s platnou stromovou strukturou.

3.2. Vybrané e-maily soukromé korespondence

Množiny E_{10} i E_{136} celé pochází z roku 2001 a e-maily v E_{10} nejsou nikterak složité a používají anglický jazyk. Proto byla zahrnuta ještě jedna množina 10 osobních e-mailů (S_{10}), přijatých na server Seznam.cz, většinou se jedná o automatické zprávy, spamy nebo řetězové dopisy, u kterých je zjevným úmyslem odesílatele rozeslání co největšímu počtu lidí nebo uveřejnění zjevně není na škodu. U ostatních e-mailů odesílatel souhlasil s jejich zveřejněním na datovém nosiči v této bakalářské práci. E-maily jsou většinou psané česky a pocházejí z let 2005-2009. Pět z nich obsahuje přílohy nebo vnořenou zprávu s přílohami a druhá polovina jsou krátké e-maily s bohatými hlavičkami, které obsahují krátké tělo, a jejichž účelem je otestovat účinnost komprese na malých souborech.

[6] Zdroj: http://arg.vsb.cz/arg/Enron_Corpus/default.aspx [4]

4. Návrh řešení komprese .eml souboru

V současné době není autorovi znám program nebo algoritmus, který by se specializoval na kompresi e-mailů. E-mail je textový soubor, nejbližší metody, které nejsou univerzální, ale specializují se na podobný druh dat, pracují s textem a jazykem, kterým je psán. Jedná se především o varianty algoritmu LZW[7] nad abecedou slov nebo slabik[8]. Na druhou stranu je e-mail textový soubor, který často nese binární data – přílohy. Komprese specifická pro daný druh dat tak musí rozlišovat jednotlivé části e-mailu. Proto je nutné v první fázi e-mail rozdělit a kompresi provádět na jednotlivé části, v co možná nejpůvodnější podobě, zvlášť. Prvním krokem algoritmu tedy je rozdělení e-mailu na logické celky.

4.1. Rozdělení e-mailu na části

E-mailová struktura je stromová a je třeba ji nějakým způsobem linearizovat. Společnou částí, přítomnou v různých částech e-mailu, jsou hlavičky, které informace o struktuře konkrétního e-mailu drží, zatímco tělo a přílohy jsou samostatné celky, které jsou na sobě nezávislé. Jako efektivní se ukázala cesta substituce, tj. nahrazení těla (přílohy) krátkým odkazem na soubor, do kterého je tělo-příloha dekódována, a způsob jeho komprese. To slije všechny hlavičky všech částí do jednoho proudu a umožní kompresi jednoho typu dat v jediném souboru, u které by se tak měl výrazně zlepšit kompresní poměr.

Kodér v první fázi tak nejprve extrahuje všechna jednodílná těla a přílohy (pouze listy ve stromové struktuře) do samostatných souborů, podle potřeby je dekóduje, zatímco hlavičky uloží do jediného souboru a zapíše do všech míst, kde začíná tělo, krátký odkaz na číslo souboru, ve kterém se vlastní tělo či příloha nachází, a zvolenou metodu jeho komprese.

V druhé fázi, případně průběžně, se všechny soubory zkomprimují. Hlavičky vlastním algoritmem, těla a přílohy již existujícími programy. Nakonec se do výsledného souboru uloží nejprve informace o verzi použité při kompresi a množině použitých metod, pak opakovaně velikost vloženého komprimovaného souboru a jeho

[7] A. Lempel, J. Ziv, T. Welch, http://en.wikipedia.org/wiki/Lempel-Ziv-Welch [5]
[8] J. Lánský (2005): Slabiková komprese, http://www.ksi.mff.cuni.cz/~lansky/SC/Diplomka.pdf [6]

obsah, na konci může být kontrolní součet. Celková režie na sestavení archívu je několik jednotek až desítek bytů navíc, toto číslo je nejvíce ovlivněno počtem a velikostí jednotlivých souborů.

Dekodér postupuje obráceně, nejprve zkontroluje, zda jsou k dispozici použité kompresní metody (pokud se jedná o externí programy), pak rozdělí archív na jednotlivé soubory, které dekomprimuje, podle informací z hlavičky případně znovu zakóduje a vloží na správnou pozici místo odkazu na příslušný soubor.

4.2. Nalezení vhodného algoritmu pro danou část

Každou část e-mailu, tedy soubor se všemi hlavičkami, tělo a jednotlivé přílohy, má smysl komprimovat různými metodami. Která z nich je ta nejlepší co do rychlosti i účinnosti, je možné určit nejméně třemi způsoby: podle typu MIME (vlastnost `Content-Type`), podle začátku souboru nebo hrubou silou, tedy vyzkoušením všech metod a zahrnutí nejlepšího výsledku. Z hlediska rychlosti je nejlepší zvolit kompresní program podle typu MIME, který určuje typ obsahu – textový, binární, nebo rovnou komprimovaný (např. obrázky, video). To však nezaručuje vždy nejlepší výsledek, také pokud chceme použít univerzální programy (např. gzip, bzip2, rar), musíme vybrat jeden nebo je testovat všechny.

Druhou možností je analýza souboru. Tato metoda má stejné výhody a nevýhody jako rozhodování dle typu MIME, akorát s tím rozdílem, že kodér nepodlehne omylu, pokud je v hlavičce uveden jiný MIME typ než ten, kterému soubor skutečně odpovídá.

Třetí, nejúčinnější, avšak časově náročnou možností je vyzkoušení všech vhodných metod a výběr nejmenšího výsledného souboru. Podle typu MIME lze v zásadě určit, zda má smysl data komprimovat (komprese gif, jpeg, mpeg, avi atd. je neúměrná ztráta času kvůli zanedbatelnému zlepšení kompresního poměru) a zda jsou textová. Komprimovat binární data lze univerzálními programy, komprimovat textová data lze kromě univerzálních programů také specializovanými programy pro kompresi textu.

Pro účely této bakalářské práce, která si klade za cíl zvýšit především účinnost komprese a srovnat jednotlivé programy, bude kompresní metoda zvolena třetím způsobem. Vzhledem k tomu, že přílohy e-mailů zřídka přesahují jednotky MB, neměla by komprese více metodami závažně zpomalit proces komprese celého e-mailu.

Seznam programů, které jsou zahrnuty v kompresním programu, jsou známé a používané univerzální programy jako gzip, rar, bzip2, které budou použity na všechny druhy dat, pro soubor hlaviček výhradně algoritmus ehc popsaný v následující kapitole.

4.3. Výběr jazyka

Protože si tato bakalářská práce klade za cíl především přehlednost kódu algoritmu na úkor rychlosti, kodér i dekodér jsou napsané v jazyce PHP. Tento jazyk byl zvolen pro bohatou výbavu vestavěných funkcí pracujících s transformací textu, jednoduchou práci s poli, regulárními výrazy, nezávislý na OS a blízký předmětu zpracování – internetovým zprávám. Weby používající PHP tak mohou kodér i dekodér využít přímo ve svém kódu při zpracování e-mailů od uživatelů. Nevýhodou tohoto jazyka je časová a paměťová náročnost.

5. Algoritmus pro kompresi hlaviček

V kapitole o struktuře e-mailu byla vybrána jednodušší a ohebnější varianta komprese struktury, kterou je hlavička e-mailu. Z minulé kapitoly máme kodérem připraven soubor, který obsahuje všechny hlavičky všech částí e-mailu bez těla a příloh, soubory s tělem a přílohami se v této kapitole zabývat nebudeme.

5.1. Datové typy, reprezentace a možné způsoby kódování

V hlavičce se dle normy vyskytuje mnoho různých datových typů. Pro účely textové komprese stačí rozeznávat několik základních datových typů, podobně jako v programovacích jazycích. Základními datovými typy tedy jsou:

- *slovo* je nejdelší takový řetězec písmen anglické abecedy, pro který platí, že jsou všechna písmena velká nebo všechna malá nebo první velké a ostatní malá. Pokud nějaké slovo tuto vlastnost nesplňuje, je rozděleno na nejdelší slova, která odpovídají této definici.
- *číslo* je nejdelší řetězec číslic. Může začínat nulou.
- *speciální znak* je znak, který není ani písmeno anglické abecedy ani číslice.
- *datum* je přesně definovaný řetězec ve tvaru `d mmm rrrr HH:MM:SS`, kde d je číselné označení dne v měsíci, mmm je třípísmenné anglické označení měsíce v roce, v rozsahu Jan–Dec, rrrr je rok v rozsahu 1970-2225 a HH:MM:SS označuje čas, vyjma dne a roku vždy s dvoucifernými hodnotami.
- *IP adresa* je řetězec ve tvaru aaa.bbb.ccc.ddd kde trojice písmen označuje číslo od 0 do 255.
- *kódovaná věta* je řetězec v in-line kódování, ve tvaru =?charset?encoding?text?= kde encoding je buď Q v případě Quoted-Printable nebo B v případě Base64, charset je použitá znaková sada.

Slovo je možné komprimovat slovníkovou metodou podobnou LZW [5]. Slova, která se v textu vyskytují vícekrát, má smysl ukládat do slovníku a do komprimovaných dat pak vkládat kód slova vypočítaný Huffmanovým[9] kódováním podle četnosti slov v textu.

[9] Canonical Huffman code, http://en.wikipedia.org/wiki/Canonical_Huffman_code [7]

Budeme při analýze a tvorbě algoritmu rozlišovat dva typy slovníků slov a dva slovníky znaků. *Zabudovaný slovník* obsahuje slova, která se v hlavičce vyskytují povinně nebo je jejich výskyt pravděpodobný. *Vlastní slovník* je slovník slov, který se vytváří při kompresi hlaviček a obsahuje slova s alespoň dvěma výskyty. *Slovník písmen* je převodní mechanismus převádějící písmena na jejich Huffmanovy kódy. *Slovník speciálních znaků* má stejnou funkci pro nealfanumerické znaky.

5.2. Frekvenční analýza

Frekvenční analýze u E_{136} a S_{10} byly podrobeny množiny:

- písmen bez ohledu na velikost
- speciálních znaků
- slov
- datových typů.

Frekvenční analýza písmen u E_{136} a S_{10} byla vstupem pro zjištění optimálních kódů znaků ve slovníku Huffmanovým kódováním.

Množina E_{136}

Frekvenční analýza písmen:

Znak	Četnost	p(znak) v %	Délka kódu
E	5699	10,32	3
O	5185	9,39	4
N	4860	8,80	4
R	4588	8,31	4
A	4441	8,04	4
T	4129	7,48	4
M	3313	6,00	4
C	3293	5,96	4
S	2798	5,06	4
I	2634	4,77	5
L	2618	4,74	5
D	1739	3,15	5
F	1261	2,28	6

Znak	Četnost	p(znak) v %	Délka kódu
Y	1242	2,25	6
X	1111	2,01	6
U	962	1,74	6
B	890	1,61	6
H	785	1,42	6
G	738	1,33	6
K	724	1,31	6
P	722	1,30	7
V	643	1,16	7
J	566	1,02	7
W	198	0,35	8
Z	63	0,11	9
Q	26	0,04	9

tabulka 1 – četnost písmen v E_{136}

Konec slova se vyskytl 12186x. Huffmanovo kódování zahrnuje i tento výskyt. Konec slova (terminátor) by měl kód tříbitový.

Frekvenční analýza speciálních znaků:

	Znak	Četnost	Délka kódu
␣	mezera	4855	2
:	dvojtečka	2461	3
↵	odřádkování	2395	3
.	tečka	2336	3
-	pomlčka	2079	3
@	zavináč	996	4
,	čárka	842	4
\	zpětné lomítko	408	5
_	podtržítko	349	5
>	větší než	162	7
<	menší než	161	7
/	lomítko	150	7

	Znak	Četnost	Délka kódu
(	levá závorka	141	7
)	pravá závorka	141	7
=	rovnítko	136	7
;	středník	136	7
“	dvojitá uvozovka	30	8
‘	apostrof	11	10
!	vykřičník	5	11
&	ampersand	4	11
⇥	tabulátor	4	11
%	procenta	1	14
?	otazník	1	14

tabulka 2 – četnost speciálních znaků v E_{136}

Znaky, které se v C_{136} nevyskytly, ale je nutné je též kódovat: **~ * $ [ ] # + | { } ^ `**

Byla jim pro účely Huffmanova kódování přiřazena četnost 1. Taktéž byl do Huffmanova kódování jednou zahrnut escape znak, po kterém následuje 8 bitů znaku, který podle normy do hlavičky e-mailu nepatří, ale připustíme jeho existenci. Zmíněné případy i escape znak mají kód délky 14 bitů.

V množině E136 bylo nalezeno 12186 slov, 997 čísel, 136 datumů a časů, žádná IP adresa, 17124 nealfanumerických znaků, 2396 konců řádků, žádné kódované věty. To znamená, že speciální znaky jsou nejčastěji se vyskytujícím datovým typem, hned po nich následují slova. Proto se označení datového typu slovo nebo znak přidělí krátké Huffmanovy kódy, zatímco ostatním typům delší.

Výsledky frekvenční analýzy slov shrnuje následující tabulka (prvních 100 nejčastějších):

pořadí	slovo	četnost
1	**X**	**959**
2	**com**	**846**
3	enron	732
4	Taylor	400
5	Mark	280
6	**Content**	**274**
7	**From**	**272**
8	**To**	**269**
9	mark	192
10	taylor	162
11	**Name**	**140**
12	M	138
13	**Type**	**138**
14	**bcc**	**136**
15	**cc**	**136**
16	**charset**	**136**
17	documents	136
18	evans	136
19	mtaylor	136
20	nsf	136
21	**plain**	**136**
22	**text**	**136**
23	thyme	136
24	All	136
25	**Date**	**136**
26	**Encoding**	**136**
27	File	136
28	Folder	136
29	Folders	136
30	Java	136
31	**Mail**	**136**
32	**Message**	**136**
33	**Mime**	**136**
34	Notes	136
35	Origin	136
36	**Subject**	**136**
37	**Transfer**	**136**
38	**Version**	**136**
39	**ID**	**136**
40	**us**	**132**
41	**ascii**	**129**
42	**bit**	**129**
43	**PST**	**86**
44	Jun	82
45	Dec	54
46	**Re**	**50**
47	**PDT**	**50**
48	**Bcc**	**44**
49	**Cc**	**44**
50	david	40
51	E	38
52	velaw	33
53	David	33
54	jones	29
55	tana	29
56	forster	28
57	Forster	23
58	Jones	22
59	Tana	22
60	bob	20
61	shults	17
62	boyd	14
63	sullcrom	13
64	R	13
65	michael	12
66	Bob	12
67	dharvey	11
68	Jeffrey	11
69	rbaird	11
70	sara	11
71	sborgman	11
72	shackleton	11
73	Online	11
74	**GTC**	**11**
75	carol	10
76	clair	10
77	debbie	10
78	justin	10
79	susan	10
80	to	10
81	Boyd	10
82	Carol	10
83	Clair	10
84	Shults	10
85	St	10
86	brackett	9
87	brent	9
88	cooper	9
89	dell	9
90	edmund	9
91	hendry	9
92	kitchen	9
93	louise	9
94	of	9
95	s	9
96	Sara	9
97	Shackleton	9
98	EOL	9
99	bp	8
100	brown	8

tabulka 3 - četnost slov v E_{136}

Slova vyskytující se v hlavičkách často, která nejsou specifická pro Enron, jsou vyznačena tučně a budou tvořit část zabudovaného slovníku. Celkem bylo 1084 unikátních slov, z toho 604 slov, která se vyskytovala vícekrát, ovšem neplatí, že se vyskytovala v každé hlavičce vícekrát, při kompresi e-mailu tak značná část z nich nebude zahrnutá do slovníku. Součet četností slov, která se vyskytovala víckrát než je počet uvažovaných hlaviček – 136x, je součet četností u prvních 13 slov, roven 4802, tj. u cca 40% slov lze předpokládat, že budou v některém slovníku. Tučných – zabudovaných slov je celkem asi 5500 výskytů (počítají se i slova s jedním výskytem). Ve vlastním slovníku bude přibližně 1900 výskytů slov (prvních 13 bez těch, která patří do zabudovaného slovníku). Znamená to, že vlastní slovník nebude velký a nejvíce slov bude v souboru jednou, nebo se budou nacházet v zabudovaném slovníku.

Hlavičky z S_{10}

Frekvenční analýza písmen:

Znak	Četnost	p(znak) v %	Délka kódu
E	863	10,51	4
A	617	7,52	4
S	584	7,11	4
I	529	6,44	4
T	519	6,32	4
O	485	5,91	4
M	470	5,73	4
R	468	5,70	4
C	452	5,51	4
N	431	5,25	5
D	340	4,14	5
L	281	3,42	5
P	240	2,92	5

Znak	Četnost	p(znak) v %	Délka kódu
U	232	2,83	5
Z	230	2,80	6
V	216	2,63	6
F	173	2,11	6
B	169	2,06	6
H	151	1,84	6
K	146	1,78	6
Y	140	1,71	6
G	130	1,58	6
X	111	1,32	7
W	92	1,12	7
J	75	0,91	7
Q	65	0,79	7

tabulka 4 – četnost písmen v S_{10}

Konec slova se vyskytl 1826x. Huffmanovo kódování zahrnuje i tento výskyt, konec slova (terminátor) by v této množině měl tříbitový kód. Kódy jsou více vyvážené. Rozdíly oproti E_{136} jsou u písemen E (+1), I (-1), N (+1), P (-2), U (-1), Z (-3), V (-1), X (+1), W (-1), Q (-2). Písmena v S_{10} tedy oproti E_{136} přibrala nejvýše jeden bit, zatímco mohla ztratit až tři. Je to dáno pravděpodobností výskytu písmen v českém jazyce. Protože je kódování pro S_{10} vyváženější a žádný kód není delší než 7 bitů, bude kódování písmen pro vytvářený slovník využívat kódy délky podle četnosti písmen v S_{10}.

Frekvenční analýza speciálních znaků:

	Znak	Četnost	Délka kódu
␣	mezera	1179	2
.	tečka	404	3
-	pomlčka	346	3
:	dvojtečka	335	3
↵	odřádkování	321	4
=	rovnítko	303	4
?	otazník	118	5
@	zavináč	115	5
(	levá závorka	87	5
)	pravá závorka	87	5
;	středník	77	6
,	čárka	75	6

	Znak	Četnost	Délka kódu
⇥	tabulátor	68	6
/	lomítko	46	6
>	větší než	42	6
<	menší než	42	6
+	plus	42	7
_	podtržítko	20	8
“	dvojitá uvozovka	18	8
[	levá hranatá	15	8
]	pravá hranatá	15	8
$	dolar	7	9
|	svislá čára	2	11

tabulka 5 – četnost speciálních znaků v S_{10}

Znaky, které se v hlavičkách S_{10} nevyskytly, ale je nutné je též kódovat: ~ * \ ‘ ! # % & { } ^ `

Byla jim také přiřazena četnost 1 a byl zahrnut escape znak. Těmto znakům vychází kódy délky 12.

Opět pozorujeme větší vyvážení rozdělení kódů. Rozdíly oproti E_{136} jsou u odřádkování (+1), rovnítko (-3), otazník (-9), zavináč (+1), kulaté závorky (-2), středník (-1), čárka (+2), tabulátor (-5), lomítko (-1), znaky nerovnosti (-1), plus (-7), podtržítko (+3), hranaté závorky (-6), znak dolaru (-5), svislá čára (-3) a nevyskytující se znaky (-2). Toto vzniklo především bohatším obsahem hlaviček, přítomností kódovaných vět, textovým označením majitele e-mailové adresy před jeho adresou a dalšími odlišnostmi. Protože je kódování pro S_{10} vyváženější a žádný kód není delší než 12 bitů, bude kódování nealfanumerických znaků pro zabudovaný slovník těchto znaků využívat kódy délky podle četností v S_{10}.

V množině S_{10} bylo nalezeno 1826 slov, 604 čísel, 46 datumů a časů, 28 IP adres, 3054 nealfanumerických znaků, 322 konců řádků, 28 kódovaných vět. Speciální znaky jsou opět nejčastěji se vyskytujícím datovým typem, po nich následují slova. Označení datového typu znak vychází kód délky 1 bit, slovo obecně 2 bity (nutné tři až čtyři pro rozlišení typu slova), tři bity pro číslo, čtyři pro datum a čas, pět pro kódované slovo, šest pro IP adresu a sedm pro označení konce souboru. To je podobné jako u rozdělení datových typů u E136, je proto další otázkou, jaký výskyt budou mít jednotlivé druhy slov.

Výsledky frekvenční analýzy slov shrnuje následující tabulka (prvních 100 nejčastějších):

pořadí	slovo	četnost
1	**cz**	**88**
2	**X**	**47**
3	seznam	40
4	**by**	**37**
5	**Received**	**37**
6	**com**	**33**
7	**from**	**24**
8	**To**	**24**
9	**with**	**20**
10	**CEST**	**19**
11	**sk**	**17**
12	**Content**	**17**
13	c	16
14	f	16
15	**id**	**15**
16	**Version**	**15**
17	b	14
18	**gmail**	**13**
19	**invoked**	**13**
20	qmail	13
21	v	13
22	d	12
23	**email**	**12**
24	stefanovi	12
25	**Reply**	**12**
26	**SMTP**	**12**
27	centrum	11
28	e	11
29	**uid**	**11**
30	**Id**	**11**
31	clean	10
32	go	10
33	result	10
34	A	10
35	**Date**	**10**
36	**From**	**10**
37	**Message**	**10**
38	**Mime**	**10**
39	**Nod**	**10**
40	**Spam**	**10**
41	**Subject**	**10**
42	**Type**	**10**
43	a	9
44	vdv	9
45	E	9
46	Smtpd	9
47	forpsi	8
48	google	8
49	mx	8
50	t	8
51	to	8
52	Seznam	8
53	**SMTPD**	**8**
54	**content**	**7**
55	david	7
56	**mail**	**7**
57	o	7
58	praha	7
59	q	7
60	s	7
61	smelhaus	7
62	srcce	7
63	tsk	7
64	**version**	**7**
65	**Fri**	**7**
66	**ESMTP**	**7**
67	**boundary**	**6**
68	**multipart**	**6**
69	R	6
70	szn	6
71	**type**	**6**
72	**www**	**6**
73	z	6
74	**Wed**	**6**
75	**PDT**	**6**
76	**charset**	**5**
77	h	5
78	i	5
79	j	5
80	l	5
81	**message**	**5**
82	**score**	**5**
83	somvprahe	5
84	spamassassin	5
85	**text**	**5**
86	volny	5
87	zzz	5
88	B	5
89	**Checker**	**5**
90	D	5
91	**Encoding**	**5**
92	H	5
93	**Priority**	**5**
94	**Status**	**5**
95	**Thu**	**5**
96	**Transfer**	**5**
97	**Tue**	**5**
98	bender	4
99	**bit**	**4**
100	**date**	**4**

tabulka 6 - četnost slov v S_{10}

Slova vyskytující se v hlavičkách často, která nejsou specifická pro obsah zpráv, jsou vyznačena tučně a budou tvořit část zabudovaného slovníku. Celkem bylo 604 unikátních slov, z toho 244 slov, která se vyskytovala vícekrát, ovšem neplatí, že se vyskytovala v každé hlavičce vícekrát, při kompresi e-mailu tak značná část z nich nebude zahrnutá do slovníku. Součet četností slov, která se vyskytovala víckrát než je počet uvažovaných hlaviček – 10x, je součet četností u prvních 30 slov, roven 635, tj. u cca 35% slov lze předpokládat, že budou

v některém slovníku. Tučných – zabudovaných slov je celkem asi 700 výskytů (počítají se i slova s jedním výskytem). Ve vlastním slovníku bude přibližně 158 výskytů slov (prvních 30 bez těch, která patří do zabudovaného slovníku). Znamená to opět, že vlastní slovník nebude velký a nejvíce slov bude v souboru jednou, nebo se budou nacházet v zabudovaném slovníku.

Proto se datovým typům přidělí kódy s délkou uvedenou před tabulkou 6, s tím, že slovo vyskytující se jednou bude mít kód délky 3 bity, slovo ze slovníku bude mít kód délky 4 (čtvrtý bit rozliší typ slovníku).

5.3. Slovo

Některá slova jsou v hlavičce povinná, jiná se vyskytují velmi často. Taková slova nemá smysl přidávat do slovníku, pro kompresní poměr je lepší, aby jejich znění znal kodér. To znamená, že budou slovníky dva (zabudovaný a vytvořený popsané v úvodu kapitoly) nebo budou oba sloučeny do jednoho. Vzhledem k tomu, že vytvářený slovník bude obsahovat jen slova vyskytující se vícekrát a ostatní slova budou kódována „in-line“, nemá smysl slovníky slučovat, naopak, zabudovaný slovník bude i jinak kódován a zabudovaná slova vyskytující se jednou budou nahrazena rovnou kódem ze zabudovaného slovníku. Vytvořený slovník tak nebude obsahovat zabudovaná slova.

5.3.1. Způsob tvorby slovníku

Navrhovaný způsob kódování je dvouprůchodový. V prvním průchodu se spočítá četnost slov, vyhodí se slova zabudovaná a slova vyskytující se jednou, pomocí kanonického Huffmanova kódování se přidělí kódy jednotlivým slovům. Slovník tak bude obsahovat informaci o celkovém počtu slov, délce kódu slova, typu (všechna velká, všechna malá, první velké) a jeho znění.

5.3.2. Seznam zabudovaných slov

V první verzi slovníku budou následující skupiny slov.

Názvy a častý obsah hlaviček: Resend, Date, From, Sender, To, Cc, Bcc, Message, Reply, In, References, Subject, Comments, Keywords, Priority, Received, from, by, via, with, for, invoked, uid, Id,

MIME: MIME, Version, Content, Type, Transfer, Encoding, Description, Disposition, text, plain, html, image, jpeg, gif, png, bmp, audio, basic, video, mpeg, application, octet, stream, msword, multipart, mixed, alternative, parallel, digest, rfc, partial, external, charset, win, iso, ascii, utf, bit, quoted, printable, base, boundary,

další častá slova: spam, checker, status, score, email, mail, cz, sk, com, net, org, SMTP, HTTP, ESMTP, SMTPD, helo, ehlo, envelope, gmail, nod, www, X,

volitelné součásti datumu (dny Mon-Sun), časové zóny. Zón je tolik, že je otázkou, zda má smysl přidávat tolik krátkých slov do slovníku. V první verzi proto pro rozsáhlost nebudou zahrnuty všechny[10], ale jen nejčastěji používané: GMT, UTC, CET, CEST, CDT, CST, EDT, EST, PDT, PST, MST, MDT.

Zvolené kódy pro zabudovaný slovník nejsou Huffmanovy – tentokrát totiž slovo v komprimovaném souboru vůbec nebude, a tak by mohlo nevhodné použití Huffmanova kódování na krátkých slovech prodloužit informaci, kterou vlastně kóduje. Proto bude délka kódu u zabudovaného slova záviset především na počtu písmen daného slova. Krátká slova mají krátký kód, zatímco dlouhá slova delší. V první verzi zabudovaného slovníku bude 107 slov. Jedno jednopísmenné (X), sedm dvoupísmenných, 35 trojpísmenných, 18 čtyřpísmenných, 14 pětipísmenných, sedm šestipísmenných, osm sedmipísmenných, deset osmipísmenných, dvě devítipísmenná, jedno desetipísmenné, čtyři jedenáctipísmenná. Nejvíce jsou zastoupená trojpísmenná slova. Slova proto rozdělíme do dvou skupin: krátká a dlouhá. Krátká budou s délkou 1-4, dlouhá s délkou větší. Jednotlivým délkám u krátkých slov můžeme přiřadit prefixy podle četností délek. Jednopísmenné slovo je jedno, může mít samo o sobě o něco delší kód než dvoupísmenné, kterých je sedm. U velkých slov lze kód určit přímo úměrný délce.

[10] Zkratek časových zón s interaktivní mapou: http://www.houkal.cz/view.php?cisloclanku=2009030001 [8]

Následující tabulka ukazuje přehled, kterým se bude řídit kódování zabudovaného slovníku:

délka slova	četnost unikátních slov	délka prefixu	logaritmus četnosti	délka kódu max.
1	1	2	0	2
2	7	3	3	6
3	35	1	6	7
4	18	4	5	9
5	14	5	4	9
6	7	7	3	10
7	8	7	3	10
8	10	7	4	11
9	2	8	1	9
10	1	9	0	9
11	4	9	2	11

tabulka 7 - kódování zabudovaného slovníku

Ve slovníku speciálních znaků budou na základě zjištěné četnosti výskytu znaky uvedené v tabulce 5 a za ní, to platí i pro délku Huffmanových kódů.

5.4. Číslo

Číslo lze kódovat dvěma způsoby: Eliasovým gama kódováním[11], které na začátek binární reprezentace čísla o jedna většího vloží tolik nul, kolik je délka vzniklého čísla v binární podobě zmenšená o jedna, nebo pevným kódováním podobným BCD či DPD[12], ovšem po dvojicích, kde se 110 možností (00-99, 0-9 na konci čísla liché délky) zakóduje do sedmi bitů s tím, že zbylých 18 možností se zkrátí (terminátor pro číslo sudé délky bude mít 3 bity). Toto kódování nazvěme 2-BCD.

[11] Eliasovo gama kódování, http://cs.wikipedia.org/wiki/Eliasovo_gama_kódování [9]

[12] Binary coded decimal, resp. densely packed decimal, http://en.wikipedia.org/wiki/Binary-coded_decimal [10]

Které kódování je lepší, ukáže následující přehled:

Číslo	Eliasovo gama kódování	Pevné kódování
0 (8 bitů)	1 (1 bit)	7 bitů
00 (16 bitů)	1+1 (2 bity)	7 bitů
1 (8 bitů)	010 (3 bity)	7 bitů
2 (8 bitů)	011 (3 bity)	7 bitů
3 (8 bitů)	00100 (5 bitů)	7 bitů
6 (8 bitů)	00111 (5 bitů)	7 bitů
7 (8 bitů)	0001000 (7 bitů)	7 bitů
14 (16 bitů)	0001111 (7 bitů)	10 bitů
15 (16 bitů)	000010000 (9 bitů)	10 bitů
31 (16 bitů)	00000100000 (11 bitů)	10 bitů
100 (24 bitů)	0000001100101 (13 bitů)	14 bitů
999 (24 bitů)	0000000001111101000 (19 bitů)	14 bitů
1000 (32 bitů)	0000000001111101001 (19 bitů)	21 bitů
16 000 (40 bitů)	0000000000000111111010000001 (27 bitů)	21 bitů
1 000 000 (56 bitů)	39 bitů	28 bitů
900 000 000 (72 bitů)	67 bitů	35 bitů

tabulka 8 - srovnání délek Eliasova a 2-BCD kódování

Z uvedené tabulky plyne, že pevné kódování dvojic číslic se nevyplatí pouze v rozsahu 0-6, 10-30, 100-126, 1000-1022.

Proto bylo důležité zjistit, nakolik se uvedené intervaly vyskytují v hlavičkách, na druhou stranu také jak častý je výskyt velkých čísel, u kterých se úspornost pevného kódování projeví nejvíce. Čísla v hlavičce se vyskytují především v řádcích Message-ID, Content-Length (dlouhá čísla), Contrent-Transfer-Encoding (7bit, 8bit, Base64), to jsou ale případy, kdy Eliasovo gama kódování nedává lepší výsledky, v datech a časech, ty se však kódují jiným způsobem jako vlastní datový typ, a individuálně v e-mailových adresách nebo předmětu zprávy. Z tohoto důvodu bude přednostně používáno pevné kódování pro dvojice číslic, pokud půjde o kompresi textu sestávajícího z číslic, a Eliasovo gama kódování pro menší čísla, jejichž významu potřebuje algoritmus rozumět jako číslu, např. označení počtu slov ve slovníku.

5.5. Speciální znaky

Speciální znaky budou kódovány pomocí zabudovaného slovníku speciálních znaků, na základě frekvenční analýzy vzorku S_{10} se stanovily Huffmanovy kódy pro jednotlivé znaky a

ty se místo nich použijí. V otázce, zda kódovat znaky zvlášť nebo jako řetězec kódů ukončený terminátorem, bylo zjištěno, že souvislá řada znaků je nejčastěji dlouhá 1-2 znaky, zřídka více, proto se vyplatí použít kratší označení datového typu speciální znak než přidávat do abecedy speciálních znaků terminátor.

5.6. Datum, čas

Datum se zakóduje po částech pevné délky. Měsíc má nejvýše 31 dní, na to vyjde 5 bitů. Označení měsíce v roce zabere 4 bity, označení roku z rozsahu 1970-2225 zabere 8 bitů. Na čas můžeme použít 86400 sekund, které má každý den, pro ně potřebujeme 17 bitů. Ty lze rozdělit i druhým způsobem, 5 bitů na hodiny, 6 bitů na minuty a 6 bitů na sekundy. Čas ještě může obsahovat informace o dni, časové zóně a její textové označení, které nemusí být pevně dáno, proto typ ukončíme po informaci o čase a zbytek, je-li přítomen, bude zakódován jako číslo a slovo.

Typ datum a čas je nejúspornější, je dlouhý vždy 34 bitů, délka původního řetězce je 20 bytů (tj. 160 bitů). Pokud počítáme označení datového typu dlouhé 6 bitů, úspora bude 75%.

5.7. IP adresa

IP adresa (IPv4) se skládá ze čtyř čísel v rozsahu 0-255 oddělených tečkou. Z toho je přímočaré kódování, které ostatně používají internetové protokoly, a to do 4 bytů, každé číslo do jednoho. IPv6 adresu, vzhledem k její poměrně variabilní reprezentaci a malému rozšíření, bude v první verzi algoritmus kódovat jako běžný text.

5.8. Kódovaná věta

Kódovanou větu ve tvaru =?charset?encoding?text?= lze v anglických e-mailech zpracovat jako běžný text, v českých e-mailech je to vzhledem k častému escapování a tím střídání datových typů poměrně nevýhodné. Celý řetězec je vhodné kódovat takto:

Na začátku je označení datového typu kódovaná věta – slouží jako počátek kódované věty. Nahrazuje znaky „=?“. Následuje text (obsah *charset*), který je kódován běžným způsobem, dále podruhé označení datového typu (nahrazuje znak „?“), za kterým následuje jeden bit – kódování, např. 0 pro Quoted-Printable a 1 pro Base64 (nahrazuje znaky „encoding?“). Hned poté následuje délka zapsaná Eliasovým gama kódováním a dekódovaný text v 8bitové formě bez dalších úprav (nahrazuje „text?=“).

5.9. Kodér

Soubor obsahující hlavičky má velikost v řádu jednotek kilobytů, zřídkakdy více. Jeho úplné načtení do paměti a vícenásobné zpracování tak zásadně nezpomaluje celkový proces komprese e-mailu. Soubor hlaviček bude komprimován ve dvou průchodech. V prvním průchodu se vytvoří vlastní slovník, odstraní se z něj slova zabudovaná a slova vyskytující se pouze jednou.

Na začátek komprimovaného souboru (.ehc) bude vložen konstantní kód, který identifikuje verzi kodéru. Následně bude Eliasovým gama kódováním zapsán počet slov vlastního slovníku.

Struktura slovníku bude sestávat z následujících částí pro každé slovo: délka Huffmanova kódu zapsaná Eliasovým gama kódováním, Huffmanův kód samotný zapsán nebude – počítá se kanonický, velikost písmen (první velké, všechna velká, všechna malá), znění slova zakódované podle tabulky 4, terminátor slova.

Struktura samotných dat bude mít vždy dvě části: označení datového typu dle rozdělení datových typů v S_{10} a reprezentace daného datového typu dle popisu ke každému datovému typu.

Na konci bude označení pro konec souboru, za ním může následovat kontrolní součet. Soubor bude před uložením doplněn o 0-7 nulových bitů tak, aby jeho délka byla v celých bytech. V první verzi kodéru i dekodéru nebude kontrolní součet přidáván ani kontrolován.

5.10. Dekodér

Dekomprese souboru s hlavičkami bude probíhat v jednom průchodu, začne kontrolou verze, rekonstrukcí (načtením) slovníku a dekódováním jednotlivých datových typů, až po dosažení označení konce souboru, a bude-li přítomen kontrolní součet, jeho kontrola.

6. Srovnání účinnosti s univerzálními algoritmy

Zde se nachází srovnání účinnosti a výkonnosti komprese jednotlivých e-mailů i jejich hlaviček samostatně.

V testování jsou zahrnuty následující programy a algoritmy:

- algoritmus EMC popsaný v kapitole 4, jež je předmětem této bakalářské práce, jeho aplikace na soubor e-mailu jako celek,
- algoritmus EHC popsaný v kapitole 5, jež je součástí řešení algoritmu EMC, použitý na dočasný soubor hlaviček,
- univerzální programy: RAR verze 3.80, bzip2 verze 1.0.5, vše na e-mail jako celek a pro srovnání také na dočasný soubor hlaviček.

V následující tabulce je přehled komprese všech dvaceti e-mailů.

		Před kompresí		EMC + EHC					RAR		bzip2	
č.	Množina	Velikost .eml souboru	Velikost hlaviček	Kompresní poměr soubor	Kompresní poměr hlavičky	Trvání soubor (s)	Trvání hlavičky (s)	Paměť pro celý soubor (kB)	Kompresní poměr soubor	Kompresní poměr hlavičky	Kompresní poměr soubor	Kompresní poměr hlavičky
1	S10	2 749	2 483	*5,16*	*4,96*	0,17	0,11	886	**4,47**	**4,40**	4,88	4,76
2	S10	1 747	1 313	4,68	**4,31**	0,14	0,06	882	**4,53**	4,64	*4,98*	*5,05*
3	S10	1 002	889	**4,53**	**4,04**	0,10	0,04	880	5,64	5,68	*5,69*	*5,73*
4	S10	2 616	2 130	*4,62*	4,36	0,15	0,07	887	**4,08**	**3,95**	4,50	*4,42*
5	S10	5 349	3 118	*4,53*	*4,71*	0,26	0,13	902	**3,41**	**3,94**	3,94	4,58
6	S10	98 662	1 914	1,35	4,17	0,26	0,06	1 298	**1,26**	**3,73**	*1,53*	*4,38*
7	S10	1 181 130	5 923	**5,27**	*4,22*	1,78	0,21	8 601	5,66	**1,82**	*5,71*	2,23
8	S10	5 815 717	2 459	**5,45**	*4,05*	6,59	0,10	26 541	*5,76*	**2,93**	5,73	3,59
9	S10	379 024	3 548	**2,89**	*4,00*	0,82	0,12	2 339	3,23	**2,55**	*3,40*	3,11
10	S10	209 679	1 717	**5,68**	4,14	0,18	0,07	1 743	*5,98*	**3,70**	5,89	*4,34*
11	E10	275 017	3 238	**5,83**	*4,24*	0,22	0,13	1 935	*5,87*	**3,07**	5,85	3,53
12	E10	14 951	9 836	*3,90*	*4,34*	0,79	0,70	1 030	**2,35**	**2,03**	2,60	2,22
13	E10	5 980	3 387	*4,45*	*4,59*	0,50	0,43	900	**3,06**	**3,28**	3,40	3,54
14	E10	167 502	1 213	1,90	**4,31**	0,28	0,08	1 667	**1,89**	4,48	*2,15*	*4,93*
15	E10	225 361	1 519	1,21	4,40	0,45	0,09	2 112	**1,20**	**4,32**	*1,29*	*4,72*
16	E10	2 037 351	2 363	**1,11**	*4,19*	1,80	0,15	10 396	1,71	**3,43**	*2,11*	4,02
17	E10	1 718 824	1 176	**1,79**	**4,17**	2,44	0,07	12 879	1,79	4,48	*2,11*	*4,90*
18	E10	81 247	465	**5,69**	**4,46**	0,15	0,02	1 402	*5,74*	*7,21*	5,70	6,83
19	E10	544 601	469	**5,45**	**4,49**	0,59	0,02	4 580	*5,47*	*7,25*	5,46	6,70
20	E10	142 398	884	1,24	**4,16**	0,30	0,05	1 600	**1,23**	5,39	*1,41*	*5,53*

tabulka 9 - srovnání kompresních metod EMC a EHC s univerzálními algoritmy RAR a bzip2

Tabulka ukazuje velikost původního .eml souboru, dočasného souboru hlaviček před kompresí, trvání procesu komprese včetně využití paměti a srovnání s metodami RAR a bzip2. Pro zvýraznění výsledků je použité podbarvení a tučné písmo u hodnoty, která je

v daném řádku a pro daná data nejnižší, kurzíva pro hodnotu, která je nejvyšší. Pokud se hodnota kompresního poměru od ostatních liší výrazně, je podbarvení silnější.

Čas byl měřen jako rozdíl funkcí microtime() volané na začátku a na konci každého hlavního PHP skriptu, čas zpracování hlaviček je zahrnut v čase zpracování celého e-mailu. Paměťová náročnost byla měřená funkcí memory_get_peak_usage() spuštěnou na samotném konci skriptu pro kompresi celého e-mailu.

6.1. Malé e-maily s velkými hlavičkami

Jedná se o prvních pět e-mailů z množiny S_{10} a dva e-maily (12, 13) z množiny E_{10}, u nichž je velikost hlaviček alespoň polovinou velikosti celého souboru.

U těchto e-mailů je patrné, s výjimkou e-mailů č. 2 a 3 z S_{10}, že kompresní poměr pro celý soubor je ze skupiny tří uvedených metod u EMC nejhorší, dokonce u souborů větších než 5 kB je rozdíl výrazný. Na druhou stranu je komprese e-mailu č. 2 metodou EMC výrazně (až o jeden bit/B) lepší, než je tomu u ostatních metod, vstupní soubor je malý (1 kB) a drtivou část souboru tvoří hlavičky, u kterých je kompresní poměr ještě lepší (až o 1,6 bitu/B oproti ostatním metodám). O něco větší soubor (č. 2) komprimuje už RAR i EMC srovnatelně.

Vybírané nejlepší metody pro tělo zprávy byly v drtivé většině případů gzip, a to pro malé soubory (do 1 kB), u větších těl se osvědčila metoda RAR. Díky tomu tělo zprávy nepřispělo k výraznému posunutí kompresního poměru.

Účinnost komprese hlaviček metodou EHC popisovanou v kap. 5 je u souborů č. 1, 5, 12, 13 znatelně horší než je tomu u univerzálních programů. Soubory č. 2 a 4 jsou komprimovány srovnatelně až lépe, zatímco nejmenší soubor ze skupiny malých e-mailů, č. 3, je komprimován oproti ostatním metodám zdaleka nejlépe.

Kompresní poměr metody EMC se pro soubory hlaviček pohybuje v úzkém rozmezí 4-5 bitů/B, zatímco ostatní metody u větších souborů hlaviček dosahují značně lepších výsledků a u menších souborů naopak horších výsledků. Výsledná velikost komprimovaného e-mailu v této skupině je dána právě účinností komprese hlaviček, proto se kompresní poměr metody EHC zásadně promítá do poměru metody EMC.

U malých e-mailů tak kompresní metoda EMC dává výsledky převážně v závislosti na velikosti souboru. Lze dobře odhadnout kompresní poměr, který bude blízký číslu 4,5 b/B a odhad velikosti komprimovaného e-mailu je pro malé soubory lineární funkcí velikostí

vstupního .eml souboru. Mezníkem výhodnosti metody EMC oproti RAR je velikost e-mailu přibližně 1,5 kB. Metoda bzip2 v tomto srovnání všeobecně nedává výrazně lepší výsledky než metoda RAR.

6.2. Velké e-maily s kódovanými přílohami

U ostatních e-mailů tvoří převážnou část velikosti souboru přílohy. Ty jsou komprimovány metodou, která dá nejlepší výsledek. Kompresní poměr metody EMC je výrazně ovlivněn převažující metodou použitou na největší přílohy. Protože jsou přílohy před kompresí dekódovány, bylo očekáváno zlepšení kompresního poměru. Z tabulky 9 je skutečně znát, u souborů 80 kB a větších, že je metoda EMC velmi často nejúčinnější. Odbouráním kódování tak dochází k malému vylepšení kompresního poměru. Pokud není nejúčinnější, je nejúčinnější metodě blízká na setiny bitu/B. To je způsobeno tím, že metoda RAR znovu komprimovala i obrázky, zatímco EMC obrázky pouze dekóduje.

K datům v tabulce je nutno uvést, že většina příloh byla komprimována metodou RAR, která se u drtivé většiny souborů ukázala jako nejlepší. U malých těl a příloh (do jednoho kB) byla často účinnější metoda gzip, ovšem toto u velkých e-mailů nemůže výrazně změnit kompresní poměr.

Soubory z množiny S_{10} č. 6 a 10 obsahuje bohaté HTML s několika obrázky, soubor č. 7 obsahuje PDF soubor, soubor č. 8 obsahuje dokumenty MS PowePoint, č. 9 obsahuje dokumenty MS Word.

Soubor č. 11 z množiny E_{10} obsahuje obrázky ve formátu jpeg, které metoda EMC nekomprimuje. Soubory č. 14, 15 obsahují dokumenty MS Word, soubory č. 16, 17 obsahují dokumenty MS Excel, č. 18 a 19 obsahuje PDF soubor, poslední e-mail obsahuje dokument RTF.

Největší e-maily obsahují jako přílohy dokumenty MS Office nebo Adobe PDF. V těchto případech, jak je vidět na souborech č. 7, 8, 9 a 16, může být odebrání kódování výraznou pomocí ke zlepšení kompresního poměru, zatímco u komprimovaných dat (obrázky) je účinnost komprese srovnatelná při kompresi libovolnou uvedenou metodou.

Komprese e-mailů s přílohami metodou EMC s výběrem nejlepšího výsledku je velmi účinná metoda, která propůjčuje účinnost známých metod na přílohy v jejich původním stavu, čímž v některých případech, zvláště u dobře komprimovatelných příloh, dochází k výraznému vylepšení kompresního poměru.

7. Uživatelská dokumentace

Kompresní algoritmy EMC a EHC jsou napsány v jazyce PHP a optimalizované pro systém Windows. Pro běh je nutné mít nainstalovaný webový server (IIS nebo Apache 2.2) a PHP preprocesor verze 5.3.0 s rozšířeními[13] array, date/time, program execution, filesystem, PHP options/info, misc., strings, variable handling (jsou standardní součástí instalace PHP), zlib a bzip2. Instalační soubory prostředí najdete na přiloženém CD v adresáři `install`, nebo je lze stáhnout z webu apache.org resp. php.net.

PHP skripty a algoritmy EMC a EHC smí být využívány pod licencí GNU/LGPL[14].

7.1. Instalace, konfigurace

Pro běh samotného programu není nutná žádná zvláštní instalace. Všechny soubory z adresáře `emc` nakopírujte do adresáře, ve kterém chcete skripty spouštět. V tomto adresáři musí mít webový server právo zapisovat, dochází totiž k vytváření dočasných adresářů při běhu programu. Totéž platí pro adresář, ve kterém se nachází e-maily, se kterými bude skript pracovat.

Konfiguraci PHP (php.ini) je nutno případně upravit tak, aby nezakazovala spouštění externích programů v adresáři se skripty a práci se soubory (funkce `eval`, `exec`, `mkdir`, `rmdir`, `copy`, `unlink`, `fopen`). Hodnotu `memory_limit` je nutné zvýšit s ohledem na velikost e-mailů, které budou zpracovávány. Orientační přehled dává tabulka č. 9 v kapitole 6 bakalářské práce. Lze také použít hodnotu -1 (bez omezení).

Doporučuje se také změnit nastavení chybových hlášení `error_reporting`, `display_errors`, `display_startup_errors` tak, aby se zobrazovala, a zahrnout i úroveň E_NOTICE.

Před změnou konfigurace PHP dosavadní funkční soubor php.ini zazálohujte. Po změně je nutné webový server (službu) restartovat. Vzorový soubor php.ini najdete na CD v adresáři `instalace/php`. Ten byl použit při vývoji skriptů, vyberte z něj potřebná nastavení.

Verze dodaná na CD používá jako jednu z kompresních metod externí program rar.exe verze 3.80, který je uložen v adresáři `emc` společně se skripty. V případě, že preferujete jiný program nebo pouštíte skripty v Unixu, je třeba v souboru config.php změnit cestu a parametry volání k metodě RAR, nebo ji vymazat.

---

[13] PHP: Extensions in alphabetical order, http://us3.php.net/manual/en/extensions.alphabetical.php [11]
[14] GNU Lesser General Public License, http://www.gnu.org/copyleft/lesser.html [12]

7.2. Spuštění

Spuštění skriptu lze provést z webového prohlížeče zadáním adresy webového serveru s cestou ke skriptu emc_compress.php resp. emc_decompress.php, s parametrem f=soubor.eml. Pokud máte skripty v adresáři emc, který je v kořeni místního webu a chcete zkomprimovat soubor email.eml, zadáte adresu:

```
http://localhost/emc/emc_compress.php?f=email.eml
```

Obdobně pro dekompresi zadejte

```
http://localhost/emc/emc_decompress.php?f=email.emc
```

Skript se pokusí soubor zkomprimovat. Pokud se to podaří, zobrazí se česky psané hlášení o úspěšném dokončení komprese, kompresním poměru, využití paměti a době jednotlivých částí komprese (hlavičky, celý e-mail). V případě, že se nezobrazí žádné jiné (chybové) hlášení, zapíše se komprimovaný soubor do stejného adresáře a stejného jména, ale navíc s příponou .emc. Dekomprese funguje obdobně, přidaná přípona u výstupního souboru je .eml. Pokud se objeví chybové hlášení, a to včetně hlášek Notice, při kompresi došlo k problému a nelze ji provést. Bývá to nesprávnou strukturou e-mailu (neočekávaný konec souboru, chybějící oddělovač ve vícedílné zprávě apod.), nebo nesprávným nastavením práv k souborům, nesprávné nastavení v php.ini, config.php, chybějící doplněk, php soubor nebo zakázaná funkce.

7.3. Chybové hlášky

Známé problémy jsou ošetřeny česky psanými chybovými hláškami, které jasně označují problém, který nastal.

„Nebyl specifikován vstupní soubor (parametr 'f')“ se objeví v případě, že zavoláte skript bez parametru f, ve kterém je očekáváno jméno souboru.

„Nelze vytvořit pracovní adresář“ se objeví v případě, že k adresáři, ve kterém je umístěn skript, nemá webový server právo zápisu nebo jsou zakázány funkce pro práci v souborovém systému..

„Soubor není platným komprimovaným e-mailem“ se objeví v případě, kdy při dekompresi z uvedeného zdrojového souboru nelze číst nebo jeho počátek neodpovídá formátu EMC.

„Metoda není známá. Zkontrolujte config.php“ se ukáže, pokud byl e-mail komprimován externí metodou, která není definovaná v konfiguračním souboru. Toto typicky nastane, pokud zkomprimujete e-mail s možností použití externího programu rar.exe a pak komprimovaný soubor přenesete na systém, který nemá metodu rar v konfiguraci.

7.4. Konfigurační soubor a kompresní metody

Paleta kompresních metod použitá kompresní metodou EMC je nastavitelná prostřednictvím souboru config.php. V něm lze definovat nové metody nebo měnit stávající, a to v rozsahu, které umožňuje PHP funkce eval, tzn. nastavení pro danou metodu se spustí jako samostatná PHP instrukce. Novou metodu je nutné zapsat do pole $metody resp. také $externi, pokud je volán externí program. Následně do polí `$komprese['metoda']['komprese']` a `['dekomprese']` je nutné vložit řetězec, který se vykoná jako samostatná instrukce a v případě interní metody vrátí v řetězci komprimovaný obsah nebo v případě externí metody zavolá komprimační program. Pro interní funkce je nutné umístit proměnnou $oznaceni_vstupu na místo, kam se vkládá vstupní řetězec. Výstupní řetězec by měl být návratovou hodnotou funkce. Pro externí programy je nutné vložit $oznaceni_vstupu na dvě místa, kam v externím příkazu patří název souboru, jednou doplněn o příponu kompresní metody (označuje výstupní soubor), a to podle pořadí parametrů volání externího programu. Interní funkce jsou během komprese volány pomocí $výstup=eval($komprese[metoda][komprese]), externí programy jsou volány stejně, ale výstupní soubor se pak načítá z dočasného adresáře.

Přílohy, jejichž MIME typ odpovídá regulárnímu výrazu definovaném v $nocompress_regexp, se program nebude pokoušet znovu komprimovat. Vyloučením některých typů souborů prostřednictvím tohoto parametru lze ušetřit čas nekomprimováním již dříve komprimovaných dat, např. obrázků (GIF, PNG, JPEG).

8. Programátorská dokumentace

Kompresní metoda EMC je sada open-source PHP skriptů, byla vyvinuta a testována na webovém serveru Apache 2.2 + PHP 5.3.0 v prostředí OS Windows XP. Jedná se o osm souborů doplněných o kompresní program rar.exe.

8.1. Moduly

Soubory emc_compress.php a emc_decompress.php jsou skripty, které pouští uživatel, aby zkomprimoval nebo rozbalil soubor s e-mailem. Cestu k němu zadává do parametru f. Po spuštění tyto soubory načtou ostatní skripty pomocí funkcí require a require_once.

Soubor config.php obsahuje definici kompresních metod, které jsou volány funkcí eval. Popis souboru config.php se nachází v uživatelské dokumentaci v poslední podkapitole.

Soubor slovniky.php obsahuje definici zabudovaného slovníku slov, písmen a speciálních znaků, datových typů, kódů měsíců a dalších převodních tabulek. Také obsahuje funkce pro zjištění kódu zabudovaného slova, čtení nebo tvorbu Eliasova kódu posunutého o libovolné celé číslo, převod celého řetězce dat mezi skutečnou binární reprezentací a posloupností textových jedniček a nul, kódování čísla do 2-BCD podoby.

Soubor lib.php je druhá knihovna funkcí pro práci s e-maily. Obsahuje funkci pro kódování Quoted-Printable, převod konkrétného znaku z binární podoby do textových jedniček a nul, funkci počítající kanonický Huffmanův kód na základě předchozího kódu a nové délky, funkci pro přesunutí dočasného souboru do proměnné a zpět, funkci pro bezpečné přejmenování souboru (pokud selhává rename() na chybu „Directory not empty“) a funkci směrování ladících hlášek, které tak lze jednoduše vypnout, zapnout nebo přesměrovat do souboru.

Soubory ehc_compress.php a ehc_decompress.php jsou součásti komprese a dekomprese e-mailu, nesou algoritmus EHC a probíhá v nich zpracování souboru hlaviček. Soubor open_files.php nese instrukce k načtení e-mailu a založení výstupních souborů.

8.2. Modul pro rozdělení e-mailu na části

Skript emc_compress.php je určen k přímému volání prostřednictvím webového prohlížeče nebo PHP konzole. Jméno souboru, který se má komprimovat, nese parametr f.

Skript založí dočasný adresář s náhodným názvem, načte e-mail do paměti, rozdělí jej na řádky a ty postupně zpracovává metodou EMC popsanou v kap. 4 bakalářské práce. Konec souboru si označí umělým oddělovačem boundary. Nejprve sestaví v hlavní smyčce dočasný soubor hlaviček, funkce komprimuj() zajistí kompresi jednotlivého těla a zahrnutí nejlepšího výsledku do výsledného archívu, funkce bin_velikost_souboru() zakóduje velikost souboru do dvou nebo čtyř bytů podle skutečné velikosti vkládaného těla/přílohy do archívu.

Mezi zpracováním e-mailu po řádcích a sestavením archívu je zavolán skript ehc_compress.php popsaný v následující podkapitole.

Při dekompresi se nejprve ověří, zda je soubor formátu komprimovaný e-mail, poté se rozbalí archív do jednotlivých souborů, po dekompresi hlaviček (volání ehc_decompress.php) je postupně zpětně sestavován soubor e-mailu, přílohy se dekomprimují a znovu zakódují podle informací v hlavičce.

8.3. Modul pro kompresi hlaviček

Soubor ehc_compress.php pracuje podle algoritmu EHC popsaném v kapitole 5 bakalářské práce, je rozdělen na funkci, která filtruje a vkládá slova do vlastního slovníku, počítá jejich četnost a četnost datových typů, následně tvoří Huffmanův binární strom řazením a přeskupováním jednotlivých částí slovníku, funkce zapis_cast_kodu() ohodnocuje vrcholy stromu podle jejich umístění a přiděluje tak kódy jednotlivým slovům. Protože se kódy přidělují kanonicky od 000… do 111…, je do slovníku v další fázi zapisována jen jejich délka. Po sestavení slovníku je soubor hlaviček procházen podruhé a kódují se jednotlivé datové typy a jejich obsah. Soubor je nakonec uzavřen vložením typu „terminátor“ a po ukončení skriptu pokračuje zpracováním ve skriptu emc_compress.php.

Soubor ehc_decompress.php načte celou binární reprezentaci hlaviček a pohybuje se v ní přes proměnnou $kurzor, obsahuje funkce pro zjištění a posunutí se o Eliasův kód, kód definovaný polem (pole ze souboru slovniky.php, např. slovník, písmeno, typ). Nejprve načte délku slovníku, vlastní slovník a následně rozkóduje datové typy.

9. Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat algoritmus využívající znalosti struktury e-mailu k jeho efektivní kompresi. Struktura e-mailu byla analyzována a byl navržen algoritmus, který se snaží datovým typům rozumět a komprimovat je efektivně podle skutečného obsahu, a to jak na úrovni celých souborů – těla a příloh, tak na úrovni uvnitř souboru – v hlavičkách.

V hlavičkách je rozeznáváno celkem šest datových typů od úrovně znaku až po úroveň řádku, známé konstrukce – zabudovaná slova se kódují v algoritmu a nikoli v souboru samotném.

Na začátku jsme měli několik předpokladů a očekávání. Výsledky komprese testovacích e-mailů ukázaly, že některá očekávání kompresní algoritmus splnil, zatímco u jiných se projevily nedostatky vyplývající z jednoduchosti algoritmu pro kompresi hlaviček.

Ad P1-P3: Použití různé reprezentace různých typů dat má za následek zlepšení kompresního poměru u velmi malých souborů s převažující hlavičkou, kdy není potřeba ukládat příliš mnoho informací o jeho obsahu. Zatímco univerzální metody mají u malých souborů horší výsledky, algoritmus pro kompresi hlaviček zachovává přibližně konstantní kompresní poměr nezávisle na velikosti vstupních dat. Na druhou stranu se kompresní poměr s přibývající velikostí textových dat u univerzálních metod zlepšuje, navržený algoritmus tak není příliš vhodný pro e-maily bez příloh větší než 2 kB.

Ad P4: Odstraněním kódování skutečně dochází ke zlepšení kompresního poměru. Protože je u velkých příloh využívána metoda poskytující nejlepší výsledek, odstraněním kódování před samotnou kompresí nemůže dojít k významnému zhoršení kompresního poměru. Negativní vliv kódování na kompresní poměr se nejvíce projevil u nekomprimovaných souborů – dokumentů Office, zatímco kódování u obrázků nemá na kompresní poměr zásadní vliv.

9.1. Možnosti dalšího vývoje

Algoritmus pro kompresi hlaviček je v první verzi poměrně jednoduchý, proto nemůže pokročilým univerzálním metodám ve většině případů konkurovat. Přesto u malých souborů dosahuje pozoruhodných výsledků a má smysl jej používat tam, kde je potřeba komprimovat velké množství velmi malých e-mailů samostatně.

Další vývoj by se mohl ubírat směrem sofistikovanější struktury pro hlavičku, která by zohlednila např. očekávané pořadí řádků nebo slov podle klíče, kterým řádek začíná.

Algoritmus pro kompresi e-mailů jako celku naopak u větších souborů často předčí univerzální metody díky znalosti jednotlivých částí e-mailu a síle těchto metod, které sám používá. Další vývoj by se mohl ubírat podrobnější analýzou přílohy, případnou dekompresí nebo dekódováním na úrovni formátu původního souboru s přílohou. V první verzi algoritmu EMC byla pro srovnání použita u hlaviček vždy metoda EHC. Malým zlepšením tak bude použití nejlepší metody i pro ty soubory hlaviček, u nichž jiné metody dávají lepší výsledky.

Literatura a odkazy

[1] IETF: Request For Comments,
http://www.ietf.org/rfc.html

[2] P. Resnick (2001): Internet Message Format, RFC 2822,
http://www.apps.ietf.org/rfc/rfc2822.html

[3] N. Borenstein, N. Freed (1996): Multipurpose Internet Mail Extensions, RFC 2045,
http://www.apps.ietf.org/rfc/rfc2045.html

[4] Enron Corpus,
http://arg.vsb.cz/arg/Enron_Corpus/default.aspx

[5] Wikipedia, A. Lempel, J. Ziv, T. Welch, LZW algorithm
http://en.wikipedia.org/wiki/Lempel-Ziv-Welch

[6] J. Lánský (2005): Slabiková komprese,
http://www.ksi.mff.cuni.cz/~lansky/SC/Diplomka.pdf

[7] Wikipedia, Canonical Huffman code,
http://en.wikipedia.org/wiki/Canonical_Huffman_code

[8] Houkal, Seznam zkratek časových zón s interaktivní mapou:
http://www.houkal.cz/view.php?cisloclanku=2009030001

[9] Wikipedia, Eliasovo gama kódování,
http://cs.wikipedia.org/wiki/Eliasovo_gama_kódování

[10] Wikipedia, Binary coded decimal, densely packed decimal,
http://en.wikipedia.org/wiki/Binary-coded_decimal

[11] PHP: Extensions in alphabetical order,
http://us3.php.net/manual/en/extensions.alphabetical.php

[12] GNU Lesser General Public License,
http://www.gnu.org/copyleft/lesser.html

Příloha – obsah CD

Součástí bakalářské práce je přiložený disk CD, který obsahuje text práce, PHP skripty, instalační balíky Apache a PHP a použité množiny e-mailů.

- /bzip2 – bzip2 verze 1.0.5 pro Windows
- /e10 – e-maily množiny E_{10}
- /emc – adresář s PHP skripty a kompresním programem rar.exe verze 3.80
- /install
    - /apache – instalační balíky Apache 2.2
    - /php – instalační balíky PHP 5.3.0
- /s10 – e-maily množiny S_{10}
- bp.pdf – tato bakalářská práce v elektronické podobě
- e136.txt – hlavičky množiny E_{136}