Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Distribuovaný systém pro detekci nevyžádaných zpráv

Plzeň 2012 Lubomír Jurečka

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 14. května 2012

Lubomír Jurečka

Poděkování

Touto cestou děkuji vedoucímu diplomové práce Ing. Pavlu Dobrému za odborné vedení a užitečné rady při psaní této práce. Rád bych také poděkoval své rodině a mým blízkým za dlouhodobou podporu a výdrž po celou dobu studia, bez kterých bych do této fáze nemohl dospět.

Abstract

A Distributed System for Spam Detection

The present thesis deals with the problem of unsolicited electronic messages (spam) on the Internet and the possible methods of the detection of these messages. The aim of this thesis is to design and to create a signature-based anti-spam filter, which is primarily intended for use on the Kerio Connect servers. The thesis presents some of the ways of the characteristic signature productions and presents its own method for determining characteristic signature, based on the frequency analysis of the message using Goertzel algorithm.

This anti-spam filter is designed and implemented as a client-server application with a distributed server and a central database. The design tended to the security of the communication between the client and the server. The implemented filter has been tested for the functionality, the reliability and the performance in an environment with the multiple clients. The quality of the proposed method of the signature production was evaluated according to the practical verification on the sample of e-mail messages (Enron corpus).

Obsah

1 Úvod

Elektronická pošta si pro svojí nenákladnost a snadné použití získala oblibu u většiny lidí spjatých s informačními technologiemi. To přilákalo především mnoho firem, které vidí v elektronické poště levný nástroj, jak prezentovat své produkty v podobě reklamy. Elektronická pošta převrátila princip marketingu, kde břemeno finančních nákladů neleží na bedrech původce reklamy, nýbrž na bedrech příjemců nevyžádaných zpráv, kteří vydávají nemalé prostředky, především v podobě ztraceného času, při třídění doručené pošty.

Jelikož infrastruktura elektronické pošty není stavěna na odepření jejího zneužití a není zde ani celosvětově uznávaná legislativa, která by upravovala rozesílání nevyžádaných zpráv (angl. „spam“), byly do procesu doručení e-mailových zpráv zařazeny doplňkové ochrany, známé jako anti-spamové fitry. A jsou to především anti-spamové filtry, které mají velkou zásluhu na tom, že je elektronická pošta stále použitelná. Některé zdroje uvádějí, že se úspěšnost průchodu spamu anti-spamovými filtry zvýšila. Mohou za to stále zlepšující se techniky generování spamových zpráv. Podle posledních trendů se spamy velmi podobají obyčejným zprávám, jsou často personifikované a přeložené do jazyka příjemce.

O doručení elektronické zprávy do schránky příjemce se starají e-mailové servery. Jedním z velkého spektra e-mailových serverů je i produkt Kerio Connect vyvíjený firmou Kerio Technologies s.r.o. V rámci zlepšení služeb chce firma nabídnou vlastní řešení filtrace nevyžádaných zpráv, čímž by zvýšila komfort a důvěru svých zákazníků. Firma v implementaci vlastního filtračního nástroje spatřuje možnost rozšíření pole působnosti.

Cílem diplomové práce je navrhnout a implementovat anti-spamový filtr, který pro každou elektronickou zprávu sestrojí charakteristickou signaturu, na jejímž základě a na základě databáze známých signatur rozhodne, zda-li je zpráva spam. V rámci diplomové práce byl navržen vlastní způsob tvorby charakteristické signatury a použit při realizaci nového anti-spamového filtru. Funkčnost tohoto filtru byla ověřena v prostředí s více klienty.

2 Nevyžádané zprávy v prostředí Internetu

Většina uživatelů internetové elektronické komunikace je obtěžováno e-mailovými zprávami, které si nikdy nevyžádali. Většinou ani netuší, kdo je jejich původní odesílatel, a marně přemýšlí, jak vůbec odesílatel získal jejich adresu. Tyto nevyžádané zprávy mohou obsahovat např. komerční sdělení, reklamy, výherní oznámení a občas spustitelné programy s obsaženými viry a trojskými koňmi. V současné době se lze setkat s podvodnými e-mailovými zprávami, získávající z adresátů přihlašovací údaje, příp. jiné citlivé údaje.

Odesílatelé e-mailových zpráv využívají anonymity, která je obsažena v infrastruktuře e-mailového systému. Data odesílatele mohou být jednoduše zfalšována a vzdáleně řízené počítače mohou být využity k rozesílání e-mailů. Nevyžádané zprávy jsou většinou rozesílány hromadně ve velkém množství. V této souvislosti se lze setkat s pojmem „spam“.

2.1 Definice

Přestože se pojem spam hojně používá, neexistuje jeho jednotná definice používaná v antispamových legislativách po celém světě. Proto jsou jednotlivé zákony týkající se spamů dosti různorodé.

Z hlediska technologického se hojně používá definice organizace Spamhaus [11]. Poskytovatelé Internetových služeb (ISP) po celém světě využívají této definice, která pracuje se slovním spojením *nevyžádaný hromadný e-mail*, anglicky „Unsolicited Bulk Email“ (UBE).

Slovo *nevyžádaný* znamená, že příjemce neudělil prokazatelné svolení k zaslání zprávy. Druhý termín *hromadný* značí, že zpráva je poslána jako část většího souboru zpráv, které mají věcně shodný obsah.

Podle organizace Spamhaus je zpráva spam, pokud je současně nevyžádaná a zaslána hromadně. Spojení *současně* tvoří neodmyslitelnou součást, jelikož regulérní zpráva může být jak nevyžádaná (počáteční zprávy, žádosti o zaměstnání), tak zaslána hromadně (odebírání zpravodajů, komunikace se zákazníky). Zda se jedná o spam není řešeno na úrovni obsahu, nýbrž na úrovni souhlasu. Z toho vyplývá, že jakákoliv nevyžádaná zpráva zaslána hromadně se považuje za spam.

2.2 Statistiky

Mnoho organizací, jako ISP, univerzity a dalších společnosti, vydalo rozličné statistiky týkajících se spamu. Přestože většina studií ukazuje skutečnost, že spamy tvoří více jak 50% z celosvětové e-mailové komunikace, přičemž nejvíce je vysílaných hosty sídlícími v USA nebo Asii, jednotlivé statistiky se liší v naměřených hodnotách. Hlavní důvody způsobující odlišnosti ve statistikách [22]:

- naměřené hodnoty jsou úzce vázány na použitou definici spamu,
- použití různých metod měření založených na průzkumu, ohlášení či technických nástrojích.

- Průzkum. Výsledky průzkumu jsou ovlivněny jak počtem podílejících se účastníků, tak jejich stanovisky. Podstatný je výběr vzorku dotázaných, dostatečně reprezentativní pro populaci, která je zkoumána. V porovnání s metodou monitorování je průzkum méně cenově nákladný a může být realizován v kratší časové době.
- Ohlášení. Při ohlášení je spoléháno na ochotu příjemců spamu hlásit příjem těchto zpráv, které jsou pak analyzovány. Metoda nemá za cíl měřit kvantitu, nýbrž identifikovat typy spamu, odesílatele spamu a určit další charakteristiky spamu. Data jsou sbírána na základě dobrovolných hlášeních a jsou subjektivní.
- Metoda založená na technických nástrojích nevyžaduje spoluúčast uživatelů. Je tedy více objektivní a přesná. Přesnost určují použité metody detekce spamu, které musí být průběžně aktualizovány podle aktuálního vývoje spamových zpráv. Technické nástroje nezaručují 100% přesnost, díky čemuž se objevují tzv. false-negative zprávy, což jsou spamové zprávy, které nebyly korektně klasifikovány jako spam, a false-positive zprávy, které byly označeny jako spam, nýbrž se jedná o normální (regulérní) zprávy. Pomocí metody založené na technických nástrojích se většinou zkoumá celkové množství spamu, typ obsahu nebo geografické umístění odesílatele.

Z hlediska lokalizace odesílatele je obtížné určit místo původu spamu, jelikož spammeři mohou využívat proxy nebo botnety, které skryjí jejich pozici.

Termínem *botnet* [27] se označuje soubor kompromitovaných počítačů připojených k Internetu. Uživatelský počítač se stává součástí botnetu spuštěním podvodného programu (malware). Ten se může do počítače dostat různými způsoby: stažením z internetových stránek, otevřením přílohy e-mailu, aktivním napadením využívajícím zranitelnosti systému nebo automatickým spouštěním programu na USB disku. Malware instaluje do počítače oběti program zvaný *bot*, který malware stáhne pomocí protokolů Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP) nebo sítě Peer-to-Peer (P2P). Bot je spouštěn při každém startu počítače a navazuje spojení s útočníkem přes Internet Relay Chat (IRC), HTTP, Domain Name System (DNS) nebo P2P síť. Útočník následně rozdává pokyny k požadovaným akcím jako jsou Distributed Denial of Service (DDoS) útoky, záměna reklam na internetových stránkách, zjišťování citlivých údajů (hesla, čísla bankovních karet), rozesílání e-mailových zpráv atd.

S ohledem na nevyžádanou poštu se lze setkat s pojmem *spambot* [14], což je počítačový program navržený ke sběru e-mailových adres z různých zdrojů internetu, jako jsou diskuzní fóra, internetové stránky apod. Spamboti mohou také vytvářet nové e-mailové účty, přes které rozesílají zprávy. Někteří boti mohou však zkoušet hádat hesla legitimních uživatelských účtů a po jejich odhalení zneužívat kompromitovaná konta.

2.2.1 Množství spamu v posledních letech

Podle statistik M86 Security[1] se poměr spamu k celkovému počtu e-mailových zpráv pohybuje okolo 73,4%. Průměrná velikost spamových zpráv je 2,4 KB. Tisková zpráva M86 Security [15] shrnuje první půlku roku 2011. Je v ní zachycen pokles počtu spamových

[1] `http://www.m86security.com/labs/spam_statistics.asp`

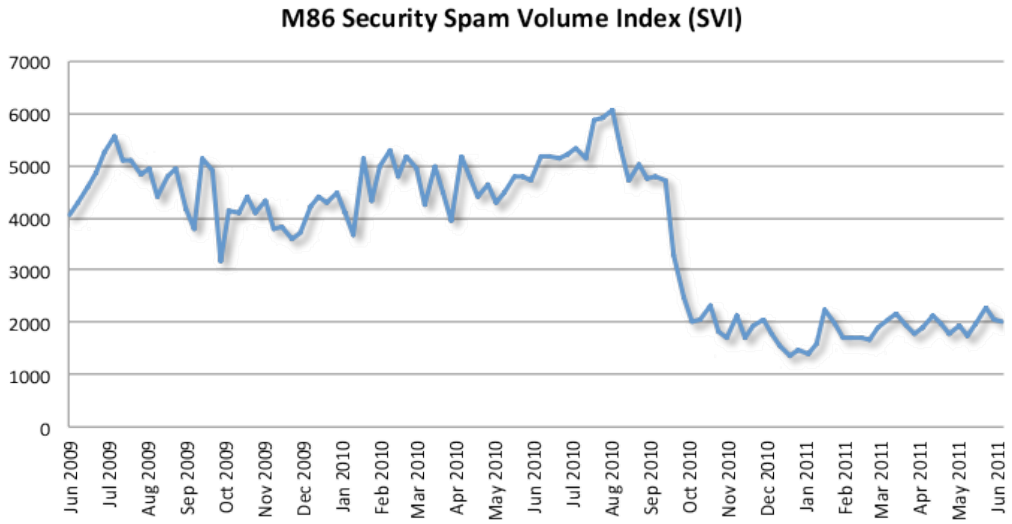
zpráv, který byl zaznamenán koncem roku 2010. Zpráva přináší několik faktorů, které vedly k onomu propadu:

- Nejvýznamnějším faktorem bylo odstavení partnerského programu Spamit.com na konci září 2010. Spamit.com byl využívaný několika spam botnety. Byl úzce spjatý s „Canadian Pharmacy“ a dalšími falešnými online lékárnami.
- Odstavení řídícího serveru pro Pushdo botnet v srpnu 2010.

Ve zprávě je popsáno více faktorů, které popisují zastavení činnosti dalších botnet sítí. Výsledkem byl pokles poměru spamu k celkovému počtu zpráv z přibližné hodnoty 90% na 77%.

Shodný propad zachycují statistiky Cisco IronPort[2] i Symantec.cloud[3]. Na obr. 2.1 je zachycen popisovaný propad. Na grafu je vynesen ukazatel Spam Volume Index (SVI), který sleduje relativní změny množství spamu zasílaného do reprezentativního vzorku honeypot domén monitorovaných organizací M86 Security.

Pojem honeypot se používá pro označení pasti, která složí k detekci, odchýlení nebo maření pokusů o zneužití informačního systému. Lance Spitzner [23] používá následující definici pro honeypot: *„A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource.“*



Obrázek 2.1: Spam Volume Index na přelomu roku 2010/11 [15].

2.2.2 Typy spamu

Spam může být rozdělen do kategorií podle cíle, jehož chtěl odesílatel spamu, neboli „spammer“, dosáhnout. Podrobný popis cílů a jejich ekonomické hledisko přináší ve své práci Guido Schryen [22].

[2] `http://www.senderbase.org/home/spam_watch`
[3] `http://www.symanteccloud.com/globalthreats/charts/spam_monthly`

Komerční sdělení

Spamové zprávy komerčního charakteru jsou označovány termínem Unsolicited Commercial E-mail (UCE). Společnosti se dívají na e-mail jako na levnou formu přímého marketingu a zároveň nástroj pro získání nových zákazníků. Komerční sdělení často nezasílají přímo společnosti, nýbrž najmutí spammeři. Kromě přímého marketingu se vyskytují v komerčních zprávách i různé formy nepřímé komerce, kde je např. doporučován nákup určitého zboží, za čímž se skrývá snaha ovlivnit cenu samotného zboží. Hojně jsou zasílané reklamní zprávy z odvětví medicíny, financí, hazardu a informačních technologií.

Nekomerční sdělení

Kromě komerčně laděných e-mailových zpráv jsou zasílány zprávy nekomerčního charakteru, jež se dotýkají např. politické, kulturní či náboženské sféry.

Podvodné zprávy

Podvodné a úmyslně zavádějící zprávy jsou zasílané spammery za účelem oklamání příjemce, který dané zprávě uvěří a reaguje na ní např. odesláním finanční částky na uvedené číslo účtu. Příkladem může být předstírání humanitární pomoci.

Lze se setkat i se specifickým typem e-mailových zpráv, vytvořeným za účelem získání osobních údajů příjemce e-mailové zprávy, jako např. čísla kreditní karty, přístupových údajů do internetového bankovnictví, atd. V této souvislosti se užívá termín „phishing“, jež vznikl obměnou slova „fishing“ (rybaření).

Phishingem [24] je míněno vytvoření e-mailových zpráv a webových stránek, jež tvoří věrohodné kopie předloh, tedy e-mailových zpráv a webových stránek známých organizací, např. bank, pro které útočník chce získat přístupové údaje uživatelů. E-mailová zpráva navádí svého příjemce na padělané webové stránky, kde je připraven formulář pro zadání všech potřebných údajů.

Hoax a řetězové zprávy

Pojem Hoax (česky kachna) je použit pro označení poplašných, nebezpečných a zbytečných řetězových zpráv. Zprávy se snaží obelstít příjemce, aby uvěřili v něco co není pravda, a navádí je k přeposlání zpráv dalším uživatelům, např. kolegům či kamarádům. Zde se naskýtá možnost sběru e-mailových adres, při kterém je adresa sběratele uvedena ve skryté kopii. Proti šíření hoaxu byly vytvořeny databáze, jež obsahují seznamy těchto zpráv. Příkladem může být webová stránka HOAX.cz[4].

Zprávy navádějící příjemce k přeposlání zprávy jsou označovány jako řetězové zprávy a mohou být zneužity pro šíření podvodného softwaru ukrytého v příloze zprávy či na webových stránkách, na které se zpráva odkazuje.

Joe job sdělení

Jedná se o hromadné rozeslání zpráv s negativním vlivem na příjemce, u kterých se snaží ovlivnit pohled a názory na odesílatele zprávy. Identita odesílatele je však úmyslně zfalšována za identitu osoby či společnosti, které má být poskvrněna reputace a dobré jméno.

[4] http://hoax.cz/cze

Spojení „joe job“ vzniklo ve spojitosti s prvním rozesláním spamu zmíněného charakteru, který provedl roznícený vlastník webových stránek, jemuž byl smazán účet z důvodu rozesílání spamu. Útok byl veden proti jeho poskytovateli webových stránek, kterým byl Joe Doll.

3 Anti-spamová opatření

Spam je již delší dobu problém, se kterým se bojuje na všech možných frontách a je jen díky různorodým anti-spamovým opatřením držen v únosných mezích. Literatura se s praxí jednoznačně shoduje v pohledu na použití anti-spamových opatření, která se mají používat doplňkově, nikoliv konkurenčně.

3.1 Legislativní opatření

S ohledem na závažnost situace a potenciální škody způsobené spamem, mnoho zemí již řeší spam pomocí legislativ. Evropská Unie (EU) přijala 12. července 2002 direktivu 2002/58/EC [17] týkající se zacházení s osobními daty a ochrany soukromí při elektronické komunikaci. Direktiva kromě záležitostí týkajících se např. určení polohy, upřesňuje opatření a ochranu proti útokům na soukromí, za něž lze považovat marketingovou komunikaci, která je často vedena formou automatického volání, telefaxu, e-mailu, SMS. *„Tyto formy komunikace ale nejsou vyžádané a pro uživatele mohou být nepříjemné. Proto se musí pro takovou formu nevyžádané komunikace nejdříve získat explicitní souhlas.“*

3.1.1 Zákon v České republice

V České republice od roku 2004 platí nový Zákon o některých službách informační společnosti (č. 480/2004 Sb.) upravující problematiku spamu, viz [13]. Zákon byl naposledy novelizován v roce 2007. Za jeho dodržováním zodpovídá Úřad pro ochranu osobních údajů.

Spam je definován jako obchodní sdělení což jsou všechny formy sdělení určeného k přímé či nepřímé podpoře zboží či služeb nebo image podniku fyzické či právnické osoby. *„Obchodní sdělení může prodejce zaslat pouze svým zákazníkům, kteří mu poskytli informovaný souhlas, v minulosti zasílání podobných sdělení neodmítli a sdělení se týkají obdobného zboží či služeb.“*

Pro získání informovaného souhlasu se musí potenciální odběratel obchodních sdělení např. registrovat či zadat osobní údaje do formuláře na webových stránkách prodejce za současného explicitního povolení zasílání těchto sdělení.

Zákon vyžaduje v každém obchodním sdělení uvést údaje o tom jak ukončit danou komunikaci.

3.1.2 Zahraničí

V rámci EU je možno postihnout odesílatele obchodních sdělení, jehož sídlo se nachází v některé ze zemí EU. Domoci se práva v ostatních případech, kdy zprávy přicházejí z jiných zemí světa, je prakticky nemožné.

Příkladem různorodosti legislativ může být zákon CAN-SPAM Act [12] Spojených států amerických (USA) z roku 2003, který nevyžaduje výslovný souhlas pro zasílání komerčních zpráv. Zákon definuje explicitní odhlášení od příjmu komerčních sdělení. V případě, že se uživatel (vlastník e-mailové adresy) odhlásí, nebude mu již zasílán žádný komerční materiál.

3.2 Technologická opatření

Vhledem k tomu, že jsou k rozesílání spamu využity technologické prostředky, není divu, že již bylo navrženo a implementováno mnoho technologických anti-spamových opatření.

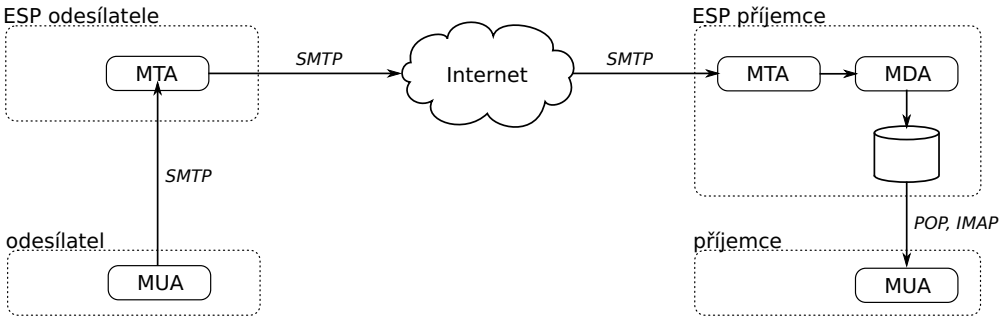
Pro snadnější orientaci v následujícím textu bude ve zkratce vysvětlen proces doručení e-mailové zprávy, při němž se na doručení zprávy podílejí následující programy:

- Mail User Agent (MUA)
  Program starající se o zpracování, příjem a odeslání elektronických zpráv uživatele. Ovládá protokoly Post Office Protocol (POP), Internet Message Access Protocol (IMAP) a Simple Mail Transfer Protocol (SMTP). MUA je součástí klientských aplikací, kterými jsou jak specializované programy (Microsoft Outlook, Thunderbird), tak webové aplikace (Gmail, Seznam), příp. konzolové aplikace (Mutt, Pine).

- Mail Transfer Agent (MTA)
  Jedná se o počítačový program, známý jako e-mailový server, realizující přenos elektronické pošty. Při přenosu rozhoduje na základě MX a A záznamů jmenných služeb.

- Mail Delivery Agent (MDA)
  MDA je počítačový program starající se o finální doručení elektronické pošty do schránek jednotlivých uživatelů.

Proces doručení e-mailové zprávy je zobrazen a obr. 3.1. Odesílatel pro vytvoření e-mailové zprávy používá poštovního klienta (MUA). Protokolem SMTP se předává vytvořená zpráva poštovnímu serveru (MTA), který zprávu převezme, zkontroluje anti-spamovými prostředky a zjišťuje, zda-li je určena pro něj. Pokud není, server přeposílá zprávu dál, podle MX a A záznamů domény, pro níž je e-mailová zpráva určena. Poštovní server ESP příjemce přebírá zprávu, provádí anti-spamové kontroly a pokud je zpráva určena pro něj a existuje záznam, kam lze zprávu uložit, je zpráva předána MDA pro doručení zprávy do schránky příjemce. Příjemce si pak zprávu vyzvedá ze schránky užitím poštovního klienta, který obvykle pro manipulaci s poštovní schránkou implementuje protokoly POP či IMAP.



Obrázek 3.1: Proces doručení elektronické pošty

Jednotlivá anti-spamová opatření se uplatňují v různých fázích doručovacího procesu. Z globálního pohledu se anti-spamová opatření mohou dělit na dvě skupiny, preventivní a reaktivní. U preventivních opatření je snaha detekovat spam dříve, než je samotná zpráva vyslána do Internetu, tzn. dochází k detekci na straně klientské aplikace (MUA

odesílatele) či na poštovním serveru ESP odesílatele. Na druhé straně stojí reaktivní anti-spamová opatření, která se snaží detekovat a filtrovat spam při příjmu zprávy na serverech ESP příjemce, příp. v klientské aplikaci příjemce. Žádoucí je blokace spamových zpráv v prvotních fázích doručovacího procesu.

3.2.1 Blokování IP

Pro přenos e-mailové zprávy mezi klientem a serverem je navázáno TCP spojení. V rámci tohoto spojení je přenášena zpráva protokolem SMTP. Jelikož IP adresa je první informací o klientu, jež je zjistitelná z TCP spojení, lze snadno odmítnou samotný SMTP přenos, v případě, že je IP adresa klienta umístěna na speciálním seznamu zakázaných odesílatelů (černý seznam), viz [22].

Černý seznam

Černý seznam, anglicky „blacklist“, je označení pro seznam blokovaných IP adres, jež byly nahlášeny v souvislosti s rozesíláním spamu.

Seznamy mohou být spravovány lokálně, kdy uživatelé e-mailových klientů, příp. administrátoři e-mailových serverů ručně vkládají či mažou IP adresy spammerů (možno i celých rozsahů IP adres). Spammeři ovšem často mění IP adresy a tak se tomuto způsobu blokace lehce vyhýbají.

Pro efektivnější a snazší využití schopností černých seznamů, vznikly organizace, jež poskytují, ať už komerčně či zdarma, seznamy blokovaných IP adres. Obvykle organizace poskytují standardizovaný přístup přes DNS, kde se pak server dotazuje na IP adresu jako na A záznam: *„19.57.227.147.sbl.spamhaus.org“*. IP adresa je blokována, je-li vrácena adresa *127.0.0.1* (loopback). Vzniká tak označení DNS Blocklist (DNSBL). Některé organizace umožňují stažení seznamů lokálně. Známým černým seznamem je např. Spamhaus Blocklist (SBL)[1].

Lze se setkat i s dočasným přidáváním IP adres do černého seznamu na základě frekvenční analýzy příchozích zpráv. Je to obrana proti náhlému zatížení serveru zprávami spammerů. Metoda se používá ve spojení s bílým seznamem, kterým se zaručí doručování legitimních zpráv, např. novinek (newsletter).

Výhodou filtrace pomocí černého seznamu je skutečnost, že přenos e-mailové zprávy může být odmítnut ještě dřív než začne, tudíž jsou ušetřeny jak systémové tak síťové zdroje. Nevýhodou je, že spammeři se vyhnou černým listům používáním IP adres jen na krátkou dobu. Na seznamu se také mohou a často vyskytují adresní rozsahy zneužitých organizací, které se poté potýkají s problémy doručování zpráv.

Bílý seznam

Opakem černého seznamu je bílý seznam, který slouží pro označení důvěryhodných IP adres. Obdobně jako u černého seznamu se může bílý seznam udržovat lokálně, příp. lze využívat služeb organizací, které spravují a poskytují své bílé seznamy.

Hodí se k předcházení problému s doručením e-mailových zpráv, u kterých je vysoká pravděpodobnost, že budou vyhodnoceny jako spam. Bílé seznamy jsou často aplikované

[1] http://www.spamhaus.org/

jako první úroveň ochrany. V případě, že zpráva pochází od důvěryhodného odesílatele, nemusí být dále kontrolována.

Šedý seznam

Šedé seznamy se výrazně podílí na úspěšné filtraci spamových zpráv. Server si udržuje tzv. triplety, které se skládají z e-mailové adresy odesílatele, IP adresy hosta snažícího se předat serveru zprávu a e-mailové adresy adresáta. Princip šedého seznamu spočívá v dočasném odmítání e-mailových zpráv, jejichž triplety server zatím nezná či ještě nenastala doba pro opakovaný přenos zprávy. V případě odmítnutí zprávy je zasláno odesílateli oznámení o odmítnutí s žádostí o opakované odeslání zprávy s časovým odstupem několika minut.

Při filtraci se využívá faktu, že většina spammerů pro zvýšení propustnosti neimplementuje obnovovací funkce. Šedý seznam představuje problém pro hosty, kteří nevyhovují specifikaci RFC 2821, jež se zabývá přeposíláním zpráv. Šedý seznam činí problémy velkým e-mailovým systémům, které používají více serverů k rozesílání zpráv, např. newsletterů. V případě, že se servery střídají v přeposílání odmítnutých zpráv, zprávy neprojdou přes šedé seznamy.

3.2.2 Analýza obsahu

Anti-spamové nástroje zabývající se analýzou obsahu [22] tvoří nedílnou součást výzbroje v boji se spamem. Při analýze obsahu jsou zkoumány jednotlivé termíny zprávy a na základě předchozích zkušeností jsou zprávy tříděny do dvou kategorií, legitimní a spamové pošty. Nástroje při analýze prohlížejí hlavičku zprávy či tělo zprávy, nebo obě části současně. Některé nástroje pracují i s přílohami. Před zahájením klasifikace se nástroje musí natrénovat pro správné rozlišení spamu od legitimních zpráv.

Nástroje analyzující obsah e-mailových zpráv se setkávají s následujícími problémy.

- Nástroje nejsou stoprocentně přesné. Chybně klasifikované zprávy se označují jako false-positive a false-negative.
- Analýza obsahu zpráv, především těla zpráv, je více náročná na systémové zdroje.
- Čím více jsou spamy podobné legitimním zprávám, tím méně jsou filtry účinné. Spammeři jsou v dnešní době poměrně úspěšní ve vytváření podob.
- Paradoxně má filtrování vliv na zvýšení zátěže pocházející z většího počtu zkoumaných zpráv. Filtrování snižuje pravděpodobnost průchodu spamu, a tak spammeři generují větší kvanta zpráv ve snaze udržet počet doručených zpráv do schránek uživatelů.

Filtrace založená na pravidlech

Filtry založené na pravidlech vyhledávají v e-mailové zprávě přednastavené hodnoty, termíny, slova či fráze, a v případě nálezu se zachovají podle požadavků uvedených v pravidlech. Definicí pravidel je umožněno vytvořit si vlastní filtr „na míru“. Seznam pravidel lze spravovat ručně, případně automaticky. Pravidla se mohou soustředit na jednotlivé položky hlavičky, případně na tělo zprávy. Příklad pravidla, které odmítá zprávy, pokud

předmět obsahuje slovo „viagra“: *„položka: Subject, typ pravidla: obsahuje řetězec, obsah: 'viagra', akce: odmítnout zprávu“.*

Filtrace založená na signaturách

Při filtraci tohoto druhu se anti-spamové nástroje nezabývají analýzou zprávy, či jejích dílčích částí, nýbrž redukují zprávu do charakteristické signatury (otisku). Používané techniky generování otisků jsou navrženy tak, aby bylo málo pravděpodobné, že legitimní zpráva bude prohlášena za spam. Díky tomu filtry mají velmi malý podíl false-positive zpráv.

Princip filtrace založené na signaturách spočívá ve vytvoření otisku zprávy a porovnání s databází již vytvořených otisků. Signatury jsou vytvářeny sofistikovanými způsoby, které dokáží čelit minoritním změnám ve zprávách, např. změna oslovení.

Anti-spamovými filtry pracujícími se signaturami zpráv jsou např. Vilpul's Razor [8] a Distributed Checksum Clearinghouses (DCC) [2]. Oba filtry jsou postaveny na distribuovaných serverech, které udržují databáze signatur.

Bayesova filtrace

Jedná se o statistický filtr, jež je založen na pravděpodobnostním Bayesově teorému. Princip filtru spočívá v analýze vstupního trénovacího vzorku spamů a legitimních e-mailových zpráv. Při analýze se zprávy rozkládají na jednotlivé tokeny, což jsou většinou jednotlivá slova zprávy. Tokenům jsou přiřazeny pravděpodobností hodnoty, jež značí, s jakou pravděpodobností se daný token může vyskytnout v nevyžádané zprávě.

Při klasifikaci e-mailových zpráv se filtr dívá na tokeny obsažené ve zprávě a na základě jejich použití stanovuje pravděpodobnost, že daná zpráva je spam.

Filtr vyžaduje natrénovat chování před zahájením klasifikace e-mailů. Trénovací množina e-mailových zpráv by měla čítat řádově stovky unikátních spamů a unikátních legitimních zpráv. Pro zvýšení efektivity filtru se pokračuje v učení i ve fázi klasifikace, jelikož se spam neustále vyvíjí, jak strukturou tak obsahem. Filtr se individuálním učením přizpůsobí terminologii používané danou organizací či uživatelem.

3.2.3 Další technologické opatření

Další technologie snižující počet doručeného spamu, využívá kontroly IP adresy vzdáleného SMTP serveru. V DNS záznamech jsou vystaveny IP adresy SMTP serverů pověřených k rozesílání e-mailových zpráv pro danou doménu. V případě, kdy se IP adresa vzdáleného SMTP serveru, jež odesílá e-mailovou zprávu, neshoduje s IP adresou vystavenou v DNS záznamech domény odesílatele, je zpráva prohlášena za spam a může být zahozena. Tento princip implementuje technologie Sender Policy Framework (SPF) [26].

Zmíněná technologie vznikla na popud skutečnosti, že spammeři falšovali v zasílaných e-mailových zprávách adresy odesílatele.

3.2.4 Techniky obcházení anti-spamových filtrů

S vývojem anti-spamových technologií se zároveň vyvíjí i samotné spamy. Odesílatelé spamů během let přišli na různé techniky, jak obejít filtry a tak s úspěchem doručit

nevyžádanou zprávu až do schránky uživatele. Mezi používané techniky patří [20]:

- omezení použití klíčových slov (jako „viagra“, „sex“),
- častá změna adresy odesílatele,
- kódování zprávy např. užitím kódování base64,
- zmatení pojmů, užití pravopisných chyb („viagra“ → „v1agra“),
- přidání HTML značek do zpráv a přímo do jednotlivých slov,
- užití obrázků místo textu,
- opakování znaků v klíčových slovech (např. „zdaaaarma“),
- skládání slov užitím Cascading Style Sheets (CSS),
- ASCII art,
- vkládání celých odstavců z literárních děl.

V článku serveru root.cz [10] je upozorněno na fakt, že posledním trendem je zasílání vcelku krátkých e-mailových zpráv podobajících se obyčejným zprávám. Obsah těchto zpráv je většinou přeložen do lokálního jazyka užitím programu pro překlad textů.

4 Způsoby vytváření charakteristických signatur

Charakteristické signatury e-mailové zprávy, nebo obecně dat, lze získat použitím standardních kryptografických funkcí, označovaných jako „hash“ funkce. Jsou jimi např. Message-Digest Algorithm (MD5) nebo Secure Hash Algorithm 1 (SHA1) .

4.1 Požadavky

Pokud by byly otisky e-mailových zpráv vytvářeny obecnými funkcemi jako MD5 či SHA1, bylo by pro odesílatele spamu velice jednoduché tyto filtry obcházet. Stačilo by k tomu vložit do každé zprávy na libovolnou pozici několik znaků, jež ve výsledku způsobí změnu otisku, takže se zpráva jeví jako unikátní.

Z toho plyne první funkční požadavek při vytváření otisku zprávy [19]. Otisky se nesmí výrazně lišit při změnách zprávy, které jsou generovány automaticky, a měly by tolerovat úmyslné škodlivé chování spammerů. Příkladem může být užití MIME kódování pro přenos zprávy za účelem vyhnutí se filtraci v průběhu doručovacího procesu. Další technikou používanou pro docílení generování odlišných signatur je využívání escape sekvencí pro zamaskování vybraných znaků HTML zprávy. Z tohoto důvodu je snahou vytvářet hašovací funkce, které jsou robustní, tzn. odolné proti zmíněných praktikám užívání různých způsobů kódování. Analýza robustních funkcí si vyžádá více času a prostředků spammerů, kteří se snaží zdokonalit své nástroje pro generování spamu.

Jelikož se e-mailové zprávy rozpoznávají pouze podle stanovených signatur, je žádoucí, aby hašovací funkce pro různé zprávy generovala různé otisky. Tím se předejde situaci, kdy legitimní zpráva bude označena jako spam, aniž by byla obsahově podobná spamu, s nímž sdílí stejný otisk. Užitím hašovací funkce s těmito vlastnostmi se získá anti-spamový filtr, který se charakterizuje velmi malým rizikem nesprávné klasifikace regulérních zpráv, tzv. false-positive zpráv.

4.2 Stávající řešení

Následující text přináší krátký popis některých anti-spamových filtrů a jejich přístupu ke generování otisků e-mailových zpráv. Jelikož spammeři analyzují chování jednotlivých anti-spamových ochran, a to nejen filtrů založených na signaturách, aby mohli upravit algoritmy generující spamy, není mnoho produktů, jež veřejně sdílí přesnější informace či specifikace.

4.2.1 Vilpul's Razor

Vilpul's Razor je distribuovaným řešením filtrace založené na signaturách. Je součástí nástroje SpamAssassin[1]. Obsahuje preprocesor pro předpřípravu textu, který umožňuje dekódování zakódovaných zpráv pomocí Base64 či Quoted-Printable (QP). Preprocesor

[1] http://spamassassin.apache.org/

ze zprávy odstraňuje HTML značky. Poskytuje několik způsobů vytváření otisku podle použité hašovací funkce: SHA1, Nilsimsa, Ephemeral. K dispozici je klient[2] psaný v programovacím jazyku Perl.

4.2.2 Pyzor

Pyzor původně vnikl jako implementace produktu Vilpul's Razor v programovacím jazyku Python, avšak s ohledem na použitý protokol a na to, že Razor servery nejsou vedeny jako Open Source, Frank Tobin postavil Pyzor na novém protokolu a poskytl ho pro volné použití[3]. Protokol pro generování otisku se skládá z následujících úkonů [7]:

- odstranění všech hlaviček e-mailové zprávy,
- pokud je zpráva delší jak čtyři řádky:
  - zahodí se prvních 20% zprávy,
  - použijí se následující tři řádky,
  - zahodí se následujících 40% zprávy,
  - použijí se následující tři řádky,
  - zahodí se zbytek zprávy,
- odstraní se všechna slova (shluky znaků oddělených prázdným místem), která jsou delší jak devět znaků,
- odstraní se vše co připomíná e-mailové adresy, URL adresy nebo HTML značky,
- vypustí se všechna prázdná místa (whitespaces),
- odstraní se všechny řádky kratší jak osm znaků a
- ze zbylého obsahu se užitím SHA1 vytvoří výsledný otisk zprávy.

4.2.3 Distributed Checksum Clearinghouse

Distributed Checksum Clearinghouse (DCC) [2]. Pracuje s dvěma způsoby stanovení otisku e-mailové zprávy za použití rozdílných funkcí kontrolního součtu: fuzzy1 a fuzzy2. Jedná se o kontrolní součty s vynecháním elementů zprávy. Programy obsažené v DCC jsou psané v programovacím jazyce C. Při zpracování zprávy jsou ignorovány prázdné řádky, bílá místa na začátku a konci řádky, některé hlavičky, oslovení, HTML značky a komentáře. Kontrolní součty jsou upravovány postupně po řádcích za použití funkce MD5.

[2] http://razor.sourceforge.net/
[3] http://pyzor.sourceforge.net/

4.3 Hašovací techniky

4.3.1 Nilsimsa

Otisk (hash) generovaný Nilsimsa technikou [6] vypadá jako klasický hash s tím rozdílem, že malá změna ve zprávě způsobí malou změnu ve výsledném otisku. Generují se 256 bitové otisky. Pro porovnání dvou otisků je použita stupnice od -128 do 128, kde nula značí shodu otisků celkem na 128 bitech. Hodnota Nilsimsy rovná 24, tj. shoda na 152 bitech, je považována za hraniční hodnotu, při jejíž překročení se považují zkoumané zprávy za podobné. Přesně řečeno: *„Any nilsimsa over 24 indicates that the two messages are probably not independently generated“*. Nástin postupu stanovení nilsimsa otisku je uveden viz [19]. Zde je uvedeno i vylepšení Nilsimsa techniky, kdy je dosaženo nižšího poměru false-positive zpráv.

4.3.2 Ephemerat signatury

Ephemeral signatury [9] jsou signatury s krátkou dobou platnosti. Princip spočívá v generování náhodných čísel, podle nichž se vybírá část e-mailové zprávy, která bude použita pro stanovení signatury. Generování náhodných čísel probíhá ve vzájemné spolupráci jednotlivých klientů. Tímto je vytvořeno pohybující se schéma generování signatur, které by mělo způsobit velké problémy spammerům v jeho „prolomení“, jelikož neví, jaká část e-mailové zprávy je podrobena kontrole.

4.4 Výběr signatury

Z dostupných způsobů vytváření charakteristických signatur e-mailových zpráv nebyl vybrán žádný, který by byl dále použit při realizaci anti-spamového filtru. Výše zmíněné způsoby tvorby signatur dostály za své existence mnoha změn, ať už to byly úpravy, opravy či inovace, čímž se bojovalo a nadále bojuje proti neustálému vývoji samotných spamových zpráv. Proto by bylo bezvýznamné snažit se o implementaci některé ze stávajících metod, přičemž by se těžko hledalo nějaké možné vylepšení. Bylo rozhodnuto vydat se cestou konstrukce vlastního způsobu vytváření signatur, který je, jak bude osvětleno dále, postaven na principech frekvenční analýzy.

Nutno poznamenat, že při výběru nebyla uvažována technika Nilsimsa, jelikož mi nebyla v době rozhodování známa.

4.5 Vlastní řešení

Další část textu se bude věnovat vysvětlením konstrukce charakteristické signatury, které vychází z jiného pohledu, než kterým bylo doposud nahlíženo na e-mailové zprávy.

Pokud se vezme v potaz, že zpráva je ve skutečnosti pouze diskrétní signál o konstantní frekvenci a jednotlivé úrovně signálu tvoří znaky z rozšířené American Standard Code for Information Interchange (ASCII) tabulky, potom lze zprávu zkoumat metodami z oboru Digital Signal Processing (DSP).

Z oboru DSP bylo zaměřeno na doménu zabývající se frekvenční analýzou. V této souvislosti se lze hojně setkat s pojmem diskrétní Fourierova transformace (DFT). Fourierova

transformace je matematická operace, která vyjadřuje časovou funkci jako funkci závislou na frekvenci (frekvenční spektrum). Pro urychlení výpočtu DFT byl vytvořen efektivnější algoritmus pojmenovaný jako Fast Fourier Transform (FFT).

Při realizaci anti-spamového filtru však nebude potřeba pracovat s celým frekvenčním spektrem zprávy, nýbrž pouze s několika málo významnými frekvencemi. Pro detekci malého počtu frekvencí by mohl být obstojně použit Goertzelův algoritmus. Srovnání a výběr vhodného algoritmu bude diskutováno v následující podkapitole.

4.5.1 Srovnání FFT a Goertzelova algoritmu

Goertzův algoritmus nahlíží na zprávu jako na digitální signál s konstantní (definovanou) frekvencí a určuje amplitudu *jediné* zadané frekvence. Při zkoumání více frekvencí se výpočet musí opakovat. Nevýhodou by mohla být snížená přesnost výpočtu při použití floatové aritmetiky.

Naproti tomu FFT nebere v potaz frekvenci vstupního signálu, nýbrž jeho délku. Počítá kompletní spektrum signálu, kde jsou promítnuty všechny harmonické složky. Stejnosměrná složka tvoří první položku spektra. Následují všechny harmonické od první, jejíž perioda odpovídá délce vstupního signálu, po N-tou. N koresponduje s délkou zprávy.

Značnou nevýhodu FFT tvoří omezení na délku zprávy, která musí být rovná mocnině dvou. Takové omezení Goertzelův algoritmus nepostihuje.

Goertzelův algoritmus [16] je pro malý počet požadovaných frekvencí méně náročný než FFT. Při úvaze zprávy o délce N znaků a k požadovaných frekvencí je složitost Goertzelova algoritmu $\mathcal{O}(Nk)$. Pro FFT se udává složitost $\mathcal{O}(NlogN)$.

Z výše uvedených poznámek byl pro následující práci vybrán Goertzelův algoritmus s ohledem na efektivnost, která vychází z principu, a nároky na délku vstupní zprávy.

4.5.2 Tvorba signatury Goertzelovým algoritmem

Algoritmus publikoval v roce 1958 Gerald Goertzel a našel si uplatnění v některých aplikacích založených na tónové detekci, například Dual-Tone Multi-Frequency Signaling (DTMF).

Kevin Banks ve svém článku [16] uvádí realizaci Goertzelova algoritmu s podrobným vysvětlením souvislostí. Algoritmus může podobně jako DFT nebo FFT pro požadovanou frekvenci zjišťovat reálnou a imaginární složku, ze kterých lze stanovit amplitudu či fázi.

Rozlišení spektra

Jak bylo poznamenáno výše, Goertzelův algoritmus nemá omezení na délku zprávy. Avšak právě délka zprávy určuje frekvenční rozlišení spektra. To znamená, že pro větší počet vzorků N, což je de facto délka zprávy, se obdrží spektrum s vyšším počtem vzorkovaných frekvencí. Rozestupy mezi jednotlivými frekvencemi jsou určeny vztahem 4.1. Hodnota vzorkovací frekvence není v tomto případě spjata s žádným aplikačním pozadím, proto byla stanovena na hodnotu 1 Hz.

$$f_{\Delta} = \frac{vzorkovaci\_frekvence}{N} \tag{4.1}$$

Jelikož algoritmus je použit pro stanovení amplitudy požadované frekvence, je žádoucí aby právě požadovaná frekvence byla celočíselným násobkem f_Δ. Tento významný faktor bude mít za následek lehkou korekci délky zprávy, jak bude vysvětleno níže.

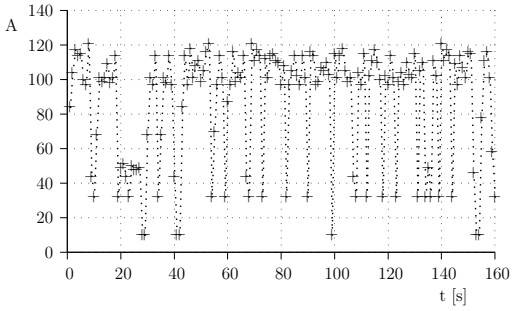
Významné složky spektra

Při zkoumání frekvenčního spektra e-mailové zprávy lze zjistit, že kromě stejnosměrné složky není ve spektru obsažena žádná další významná složka. Pokud by byla použita k porovnávání e-mailových zpráv pouze stejnosměrná složka, bylo by dosaženo vysokého počtu shodných zpráv, které by ve skutečnosti nebyly vůbec obsahově podobné.

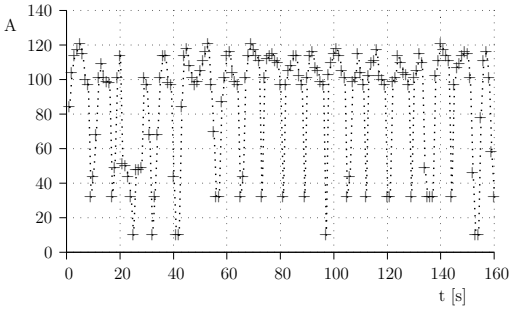
K tomu, aby bylo možno ze spektra zprávy obdržet více charakteristických složek, musí být zpráva určitým způsobem předem upravena. Možnou úpravou je přeuspořádání znaků ve zprávě podle ASCII hodnot tak, aby se zavedl ve výsledném digitálním signálu určitý řád. Pokud budou znaky postupně řazeny v malých blocích s tím, že v lichých (resp. sudých) blocích se znaky seřadí vzestupně (resp. sestupně), dosáhne se jisté harmonizace signálu. Harmonické složky signálu budou figurovat na frekvencích určených rovnicí 4.2.

$$f_h = h \cdot \frac{vzorkovaci\_frekvence}{2 \cdot velikost\_bloku}, \quad h = 1, 2, \ldots, velikost\_bloku \tag{4.2}$$

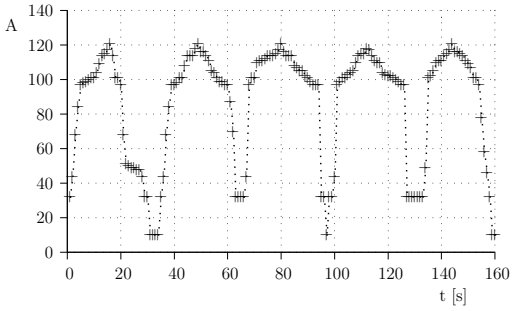
Předešlou myšlenku vystihuje obr. 4.1, kde jsou zachyceny celkem tři grafy znázorňující digitální signály originální zprávy (a) a zpráv s přeuspořádáním podle bloku o velikostech 4 (b) a 16 (c) vzorků. Přestože se jedná o diskrétní signály, jsou jednotlivé body pro lepší názornost a orientaci spojeny tečkovanou čarou. Písmenem A je v grafech označena amplituda vzorku, neboli ASCII hodnota vzorku.



(a) originální zpráva



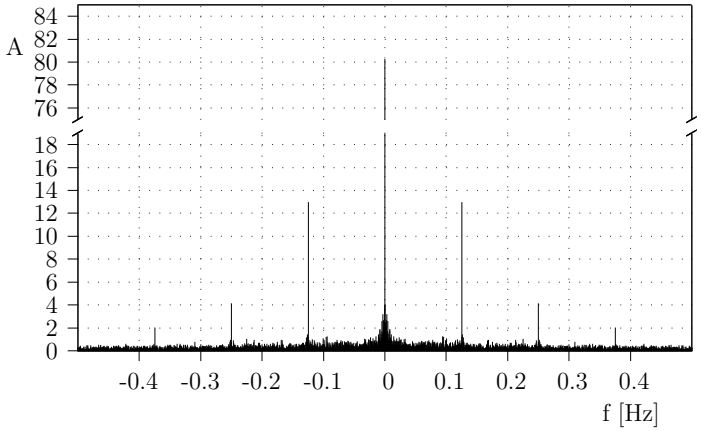
(b) s velikostí bloku 4 vzorky



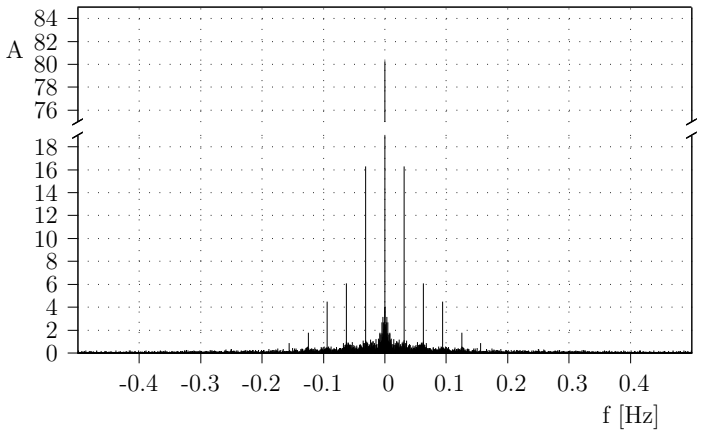
(c) s velikostí bloku 16 vzorků

Obrázek 4.1: Grafické znázornění upraveného digitálního signálu

Na obr. 4.2 jsou znázorněny frekvenční spektra upravených digitálních signálů užitím bloků o velikostech 4 (a) a 16 (b) vzorků. Z důvodu velkého nepoměru mezi střední a ostatními složkami spektra, byly grafy přerušeny v ypsilonové ose. Grafy znázorňují závislost amplitudy na frekvenci.



(a) s velikostí bloku 4 vzorky



(b) s velikostí bloku 16 vzorků

Obrázek 4.2: Frekvenční spektra upravených signálů

Výrazné frekvence jsou pro různé velikosti bloků odlišné. Souvislost mezi frekvencemi a velikostí bloku vyjadřuje vztah 4.2, kde symbol h vybírá harmonickou složku. Je žádoucí, aby Goertzelův algoritmus počítal amplitudu pro frekvence, které jsou obsaženy v rozlišení spektra. To se zabezpečí úpravou počtu vzorků signálu. Počet vzorků musí být bezezbytku dělitelný dvojnásobkem velikosti bloku, použitého k přeuspořádání vzorků, viz vztah 4.3.

$$N = 2k \cdot velikost\_bloku, \quad k \in \mathbb{N} \tag{4.3}$$

Přizpůsobení délky zprávy

Zůstává tedy otázkou, zda-li zprávy zkracovat nebo naopak prodlužovat vhodnou výplní na požadovanou velikost. Pro delší zprávy se jeví jako vhodná první varianta, která nebude mít takový dopad na zkoumané amplitudy. Při úvaze, že se dlouhé zprávy nebudou

zpracovávat celé, bude způsob zkracování zprávy takto identický. Délka maximální části, která bude z e-mailové zprávy zpracovávána, je s ohledem na statistiky spamu stanovena na 2048 znaků.

Ovšem zkrácení kratších zpráv, kde se velikost zpráv pohybuje na úrovni velikosti bloků používaných k uspořádání vzorků, může mít za následek, že nepůjdou stanovit rozumné výstupní hodnoty. Proto kratší zprávy, než dvojnásobek největšího použitého bloku, budou rovnou vyřazeny ze zpracování.

Zhodnocení řešení

Pro praktické zhodnocení navržené konstrukce otisku zpráv a výběru vhodných parametrů bylo vytvořeno testovací prostředí, které umožnilo měnit následující parametry:

- počet použitých bloků,
- počet znaků, definující velikost bloku,
- použité složky (stejnosměrná + 3 hramonické složky),
- procentní povolená diference amplitud.

Při testování byl využit vzorek spamových zpráv [18], který čítal po úpravě 8542 obsahově rozdílných e-mailových zpráv.

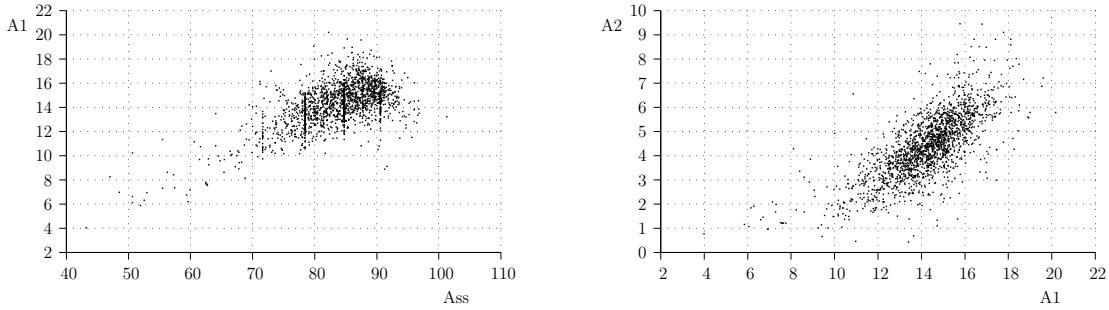
V tab. 4.1 je uvedena statistika testování, které bylo prováděno pro signatury tvořené z dvou bloků o velikostech 4 a 16 znaků. První sloupec tabulky tvoří číselník testovaných případů. Druhý sloupec přináší povolenou procentuální diferenci hodnot signatur, při níž jsou zprávy prohlašovány za podobné. Třetí a čtvrtý sloupec informuje o počtu použitých harmonických složek a zda-li byla použita stejnosměrná složka. Další sloupce tabulky tvoří naměřené hodnoty. Hodnota úspěšnosti udává, z kolika procent došlo ke správnému spárování obsahově podobných e-mailových zpráv. Ve výsledcích se objevují vysoké počty nalezených dvojic, což je způsobeno tím, že se použitý vzorek skládá z velkého množství podobných zpráv.

#	Diference [%]	Počet harm. složek	Použití ss. složky	Úspěšnost [%]	Počet dvojic	Počet chyb
1	0,5	3	ne	100	1532	0
2	0,5	3	ano	100	1244	0
3	1,0	3	ano	100	3149	0
4	1,0	3	ne	99,7941	3400	7
5	0,5	2	ano	99,7557	1637	4
6	0,5	2	ne	97,8776	2026	43
7	1,0	2	ano	97,1708	3676	104
8	1,0	2	ne	81,7449	4711	860

Tabulka 4.1: Naměřené hodnoty pro různé parametry signatur

Z obdržených výsledků pro následnou realizaci anti-spamového filtru se vybrala jako vhodná hodnota povolené diference 0,5%. Pro jednotlivé bloky budou stanoveny amplitudy tří harmonických složek, bez stejnosměrné složky. Její absence je vysvětlena v následujícím odstavci. Celkem tedy bude signatura tvořena ze šesti hodnot.

Při studii výsledků zobrazených v trojrozměrném grafu, kde dimenze tvořily postupně amplitudy stejnosměrné, první a druhé harmonické složky, bylo zjištěno, že stejnosměrná složka má velký vliv na shlukování. Na obr. 4.3 je tento jev zachycen v grafické podobě (použity dva 2D grafy) a byl zaznamenán i v přesnosti testových případů 4 a 7, kde se v obou případech pracovalo se třemi složkami. Proto nebude amplituda stejnosměrné složky brána v potaz.



(a) stejnosměrná a první harmonická složka (b) první a druhá harmonická složka

Obrázek 4.3: Umístění signatury v 2D prostoru

Pro výběr vhodných velikostí bloků, bylo provedeno měření, při kterém byla konstantní povolená diference, rovná hodnotě 0,5%, za současného využití tří harmonických složek bez stejnosměrné složky. Výsledky měření jsou zachyceny v tab. 4.2. Je patrné, že samotná velikost bloků nemá až tak výrazný vliv na obdržené výsledky. Podstatný rozdíl je zaznamenán v případech, kdy je použit pouze jeden blok. Pro následnou realizaci anti-spamového filtru byla vybrána varianta stávající se z velikostí bloků 4 a 16 znaků.

Velikost 1. bloku	Velikost 2. bloku	Úspěšnost [%]	Počet dvojic	Počet chyb
4	16	100	1631	0
8	16	100	1617	0
8	9	100	1594	0
4	10	100	1578	0
8	32	99,9461	1857	1
6	16	99,87	1539	2
4		82,1775	2682	478
8		78,1553	3090	675
6		77,4969	3244	730

Tabulka 4.2: Naměřené hodnoty pro různé velikosti bloků

Uvedená technika vytváření signatury užitím Goertzelovým algoritmem má několik nedostatků. Signatury jsou dosti náchylné na změny ve zprávách, ve smyslu přidání či odebrání znaků ze zprávy. Se záměnou znaků není problém, pokud se nejedná o změny s větším skokem ASCII hodnoty, např. záměna písmene za číslo. Dalším nedostatkem je skutečnost, že zprávy, jejichž obsahy se navzájem vůbec nepodobají, mohou být zobrazeny do velice podobných signatur. Tím se zvyšuje nežádoucí poměr false-positive zpráv.

5 Návrh systému

V této kapitole je věnována pozornost podrobnému návrhu systému a jeho jednotlivých komponent. Z podstaty filtrace založené na signaturách vyplývá, že jednotlivé e-mailové servery si musejí nějakým způsobem sdílet informaci o vytvořených signaturách. Snadným způsobem sdílení takové informace je použití centrálního serveru, který sbírá od svých klientů e-mailové signatury, vede si statistiky a na žádost klientů odpovídá s jakou pravděpodobností se jedná o hromadně zasílanou e-mailovou zprávu. Navrhovaný systém je realizován jako systém s architekturou klient-server.

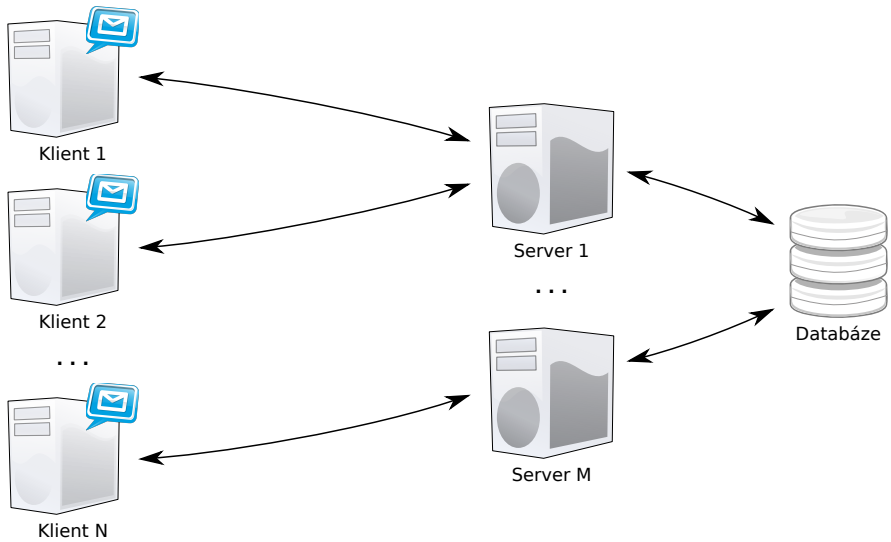
5.1 Požadavky

Aplikace pro filtrování e-mailových zpráv, navrhována v této práci, by měla být nasazena do e-mailového serveru Kerio Connect, dále jen „e-mailový server“, firmy Kerio Technologies s.r.o.

Při navrhování systému se musí brát zřetel na bezpečnost a bezpečnostní rizika při komunikaci klientů se serverem a s tím spojenou autentizaci a autorizaci klienta při přenosu jednotlivých zpráv, bude-li potřeba.

5.2 Architektura

Navržený a implementovaný systém je postaven na architektuře klient-server. Klient se stará o vytváření charakteristických signatur e-mailových zpráv, server shromažďuje signatury a čítá četnosti podobných výskytů. Program serveru je realizovaný tak, aby mohlo běžet najednou více jeho instancí. Za servery je ukryta centrální databáze, ve které jsou udržována všechna potřebná data, především e-mailové signatury. Popisovanou architekturu vystihuje obr. 5.1.



Obrázek 5.1: Architektura systému

5.2.1 Klient

Klient je navržený jako knihovna, sloužící ke zjištění četnosti výskytu e-mailových zpráv s podobnou charakteristickou signaturou. Podrobný popis rozhraní knihovny přináší kap. 6.3.

Jelikož knihovna bude používána po celou dobu běhu e-mailového serveru, je od ní požadována schopnost dlouhodobého běhu a řízení chybových stavů. Za jakýchkoliv podmínek klient vrací e-mailovému serveru smysluplná data. Klient řeší problémy, způsobené nepříznivými podmínkami, dočasným výpadkem. V době výpadku odpovídá na požadavky chybovým kódem. Za nepříznivé podmínky je např. považován výpadek serveru, ztráta konektivity, atd.

5.2.2 Server

Server se skládá jak z programu obsluhujícího požadavky klientů, tak webového serveru, který slouží pro přihlášení klienta do systému.

Webový server nemusí být nasazen na všech potenciálních serverech systému. Musí být ovšem dostupný pro všechny klienty. Webový server pracuje s centrální databází, kde eviduje nově příchozí klienty, a stará se o udržování čistoty databáze. Z databáze pomocí automaticky spuštěných skriptů maže starší záznamy porovnáváním časových značek.

5.3 Komunikační vrstva

Komunikace mezi serverem a klientem je realizována navrženou komunikační vrstvou.

5.3.1 Výběr protokolu

Pro realizaci přenosu dat mezi klientem a serverem bylo vybíráno mezi následujícími protokoly: HTTP(S), TCP, UDP, DNS.

HTTP Hypertext Transfer Protocol (HTTP) trvoří protokol aplikační vrstvy a slouží k výměně dat mezi webovým serverem a prohlížečem. Je přenášen protokolem TCP. Jedná se o bezestavový protokol.

HTTPS Hypertext Transfer Protocol Secure (HTTPS) je zabezpečenou verzí protokolu HTTP, kde před samotným přenosem dat je navázáno zabezpečené spojení. Přenášená data jsou šifrována pomocí SSL nebo TLS. Díky certifikátu lze ověřit identitu serveru.

TCP Transmission Control Protocol (TCP) tvoří protokol pro spolehlivé doručování dat ve správném pořadí. Pro přenos je navazováno spojení mezi koncovými uzly. Při použití lze řešit otázky trvanlivosti navázaného spojení, zda-li navazovat spojení pro každý požadavek nebo využít spojení k přenosu více požadavků.

UDP UDP neboli User Datagram Protocol je určen pro přenos dat bez navázání spojení. Odeslaná data mohou dojít mimo pořadí, duplicitně nebo vůbec. Protokol UDP garantuje jediné, pokud jsou data přijata, jsou korektní. Pro zabezpečení doručení

je potřeba realizovat vlastní detekci ztrát paketů. Pro zabezpečení přenosu dat proti podvrhu a dalším útoků je třeba data šifrovat.

DNS Domain Name System (DNS) je protokolem aplikační vrstvy. Dobře prochází přes firewally. Chybí ovšem autentizace, která by se musela řešit zřejmě pomocí bílých listů s IP adresami. IP adresa by se pak ověřovala použitím jiného protokolu, např. HTTPS.

Pro komunikaci mezi klientem a serverem byl vybrán protokol UDP, z důvodu malého množství přenášených dat v jednotlivých paketech a vyloučení nutnosti spravování velkého počtu otevřených soketů na serveru. Nad UDP je vytvořena vrstva pro zabezpečení doručení dat a zároveň k šifrování přenášených dat. K ověření identity klienta, identity serveru a výměně prvotních dat (identifikátor klienta, šifrovací klíč a další) je použit protokol HTTPS.

5.3.2 Schéma komunikace

Klient pro přihlášení do systému navazuje zabezpečené spojení s webovým serverem. Metodou POST předává serveru licenční číslo a verzi, kterou klient podporuje. Pokud je licenční číslo obsaženo v systému, server vrací inicializační informace, jako jsou identifikátor klienta, sekvenční číslo, symetrický klíč, velikost okna pro paralelizaci dotazů, seznam serverů seřazený dle vytížení a maximální doba spojení, po jejíž vypršení se klient musí znovu přihlásit do systému. V případě, kdy licenční číslo není korektní, webový server vrací chybový kód s časovým omezením pro opětovný pokus přihlášení, kdy klient může opakovat přihlášení až po vypršení časové prodlevy definované v odpovědi serveru.

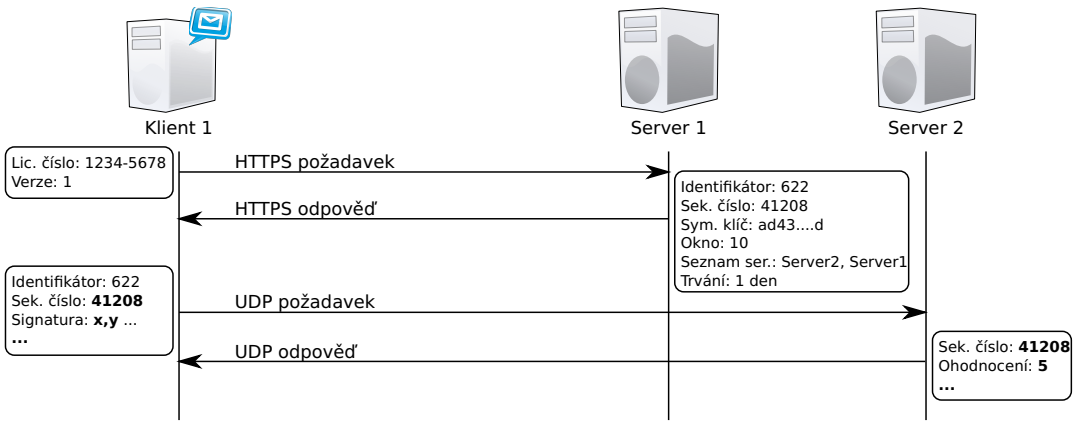
Po přihlášení již klient zasílá požadavky na server UDP protokolem. Struktura přenášených dat je popsána v následující kapitole. Klient zasílá požadavky na server, který obdržel v seznamu serverů na prvním místě. Dojde-li k překročení definované hranice pro maximální povolené zpoždění příjmu odpovědi, klient začne posílat požadavky na další server v seznamu. Tento postup může opakovat ještě jednou. Další případná změna serveru již nenastane a klient se musí opět přihlásit do systému, kdy obdrží nové identifikační údaje s novým seznamem serverů.

Komunikace mezi klientem a serverem je typu požadavek-odpověď. Komunikaci iniciuje klient, který se i stará o kontrolu doručení zprávy. Schéma komunikace je zachyceno na obr. 5.2, kde tučně zvýrazněné přenášené hodnoty jsou šifrovány symetrickým klíčem.

5.3.3 Paralelizace požadavků

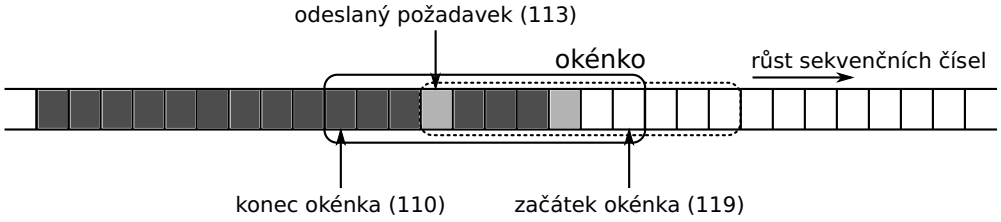
Oba programy, jak klient tak server, při přenosu dat pracují s tzv. klouzavým okénkem, díky němuž může klient vyslat současně několik požadavků. Implementované chování klouzavého okénka je až na detaily shodné s protokolem Selective Repeate. V současné implementaci je velikost okénka ustálená na hodnotě deseti zpráv. V průběhu vykonání programů se velikost okénka nijak nemění. Jelikož je komunikace mezi klientem a serverem typu požadavek-odpověď, může se odpověď pokládat za potvrzovací zprávu ACK.

Klient posouvá konec okénka až na pozici prvního požadavku, na který zatím nepřišla odpověď, případně na první volnou pozici, když se nečeká na žádnou odpověď. Na obr. 5.3 je znázorněné okénko klienta, které bude v následujícím okamžiku posunuto o tři místa



Obrázek 5.2: Schéma komunikace klienta se servery

doprava na první požadavek se sekvenčním číslem 113, který čeká na odpověď. Tmavě šedá pole znázorňují obsloužené požadavky, světle šedá pak požadavky zatím bez odpovědi. Pokud odpověď nepřijde do stanoveného časového limitu, požadavek je opakovaně odeslán na server.



Obrázek 5.3: Klouzavé okénko klienta

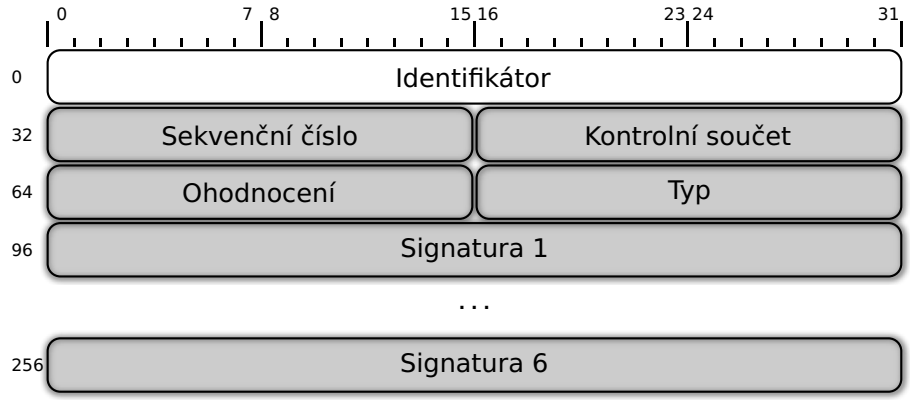
Server manipuluje obdobně s klouzavým okénkem, jen s tím rozdílem, že začátek okénka je držen na nejnovějším příchozím požadavku. Tak činí proto, aby rozeznal požadavky, které klient opakovaně odeslal. Na tyto požadavky server odpoví, ale neaktualizuje statistiky signatur. Požadavek se sekvenčním číslem menším, než na které ukazuje konec okénka, jsou zahozeny. Naopak požadavky se sekvenčním číslem větším, než na které ukazuje začátek okénka, jsou zpracovány následujícím způsobem. Pokud je sekvenční číslo požadavku menší, jak sekvenční číslo začátku okénka zvýšené o trojnásobek velikosti okénka, je požadavek zpracován a začátek okénka posunut na přijaté sekvenční číslo. V opačném případě je požadavek zahozen. Násobení okénka koeficientem rovným třemi vychází z možností klienta, kdy bez nutnosti opětovného přihlášení do systémů může postupně využít tři servery pro zasílání požadavků. Nebudou-li první dva servery dostupné, bez násobení třemi by se požadavky zasílané na třetí server zahazovaly.

5.3.4 Návrh struktury paketů

Pakety zasílané mezi klientem a serverem mají dvojí podobu. První paket tvoří požadavek zasílaný na server a jeho struktura je zachycena na obr. 5.4.

Druhý paket, viz obr. 5.5, slouží jako odpověď ze serveru. Zašedlé části paketů jsou před odesláním šifrované symetrickou šifrou, viz kap. 5.3.6. Šifrovaná část paketu požadavku tvoří 32 bajtů, resp. odpovědi 16 bajtů.

Struktura požadavku

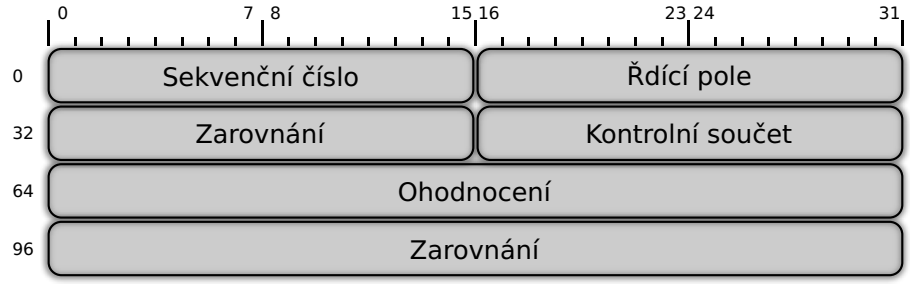


Obrázek 5.4: Struktura paketu požadavku

- **Identifikátor** (32*b*) Identifikátor slouží k identifikaci klienta. Serveru slouží tato položka k nalezení symetrického klíče, kterým dešifruje zbývající obsah zprávy.
- **Sekvenční číslo** (16*b*) Přítomnost sekvenčního čísla je vyžadována z několika podstatných důvodů. Díky sekvenčnímu číslu může klient párovat jednotlivé požadavky a odpovědi. Server si udržuje bitovou mapu posledně zodpovězených požadavků, podle které pozná, že klient opakuje požadavek. V tomto případě server neaktualizuje statistiky, pouze odpoví. Pomocí sekvenčního čísla lze bojovat proti snahám znehodnotit vedené statistiky opakovaným přenosem paketů, které zachytil útočník.
- **Kontrolní součet** (16*b*) Ačkoliv hlavička UDP paketu obsahuje kontrolní součet a UDP protokol zaručuje příjem pouze bezchybně přijatých dat, je zde kontrolní součet rovněž uveden a to z důvodu kontroly správného dešifrování dat. V případě podvrhnutí dat bude přijatý paket při kontrole kontrolního součtu zahozen.
- **Ohodnocení** (16*b*) Dosavadní spamové ohodnocení e-mailové zprávy vytvořené filtry poštovního serveru. V současnosti na ohodnocení není brán zřetel.
- **Typ** (16*b*) Typ zprávy, který je zjištěn z hlavičky e-mailové zprávy, konkrétně z položky *Content-type*.
- **Signatura** ($6 \cdot 32b$) Vytvořené charakteristické signatury e-mailové zprávy.

Struktura odpovědi

- **Sekvenční číslo** (16*b*) Sekvenční číslo je přítomno i v paketu odpovědi pro spárování s požadavkem.
- **Řídící pole** (16*b*) Řídící pole slouží pro řízení klienta serverem. Server by mohl nastavením patřičného příznaku vynutit přepojení klienta, příp. omezit paralelizaci požadavků. V současnosti se možnosti řídících příznaků nevyužívají.



Obrázek 5.5: Struktura paketu odpovědi

- **Zarovnání** (16*b*) Nevyužitý prostor paketu.
- **Kontrolní součet** (16*b*) Obsažen ze stejného důvodu jako v paketu požadavku.
- **Ohodnocení** (32*b*) Celkový počet výskytů e-mailových zpráv s podobnou charakteristickou signaturou.
- **Zarovnání** (32*b*) Zarovnání paketu na délku 16 bajtů, z důvodu použití blokové šifry o velikosti bloku 16 bajtů, viz kap. 5.3.6.

5.3.5 Spolehlivost

Použitý protokol UDP není spolehlivý. Pakety se mohou ztrácet, přicházet ve špatném pořadí, nebo přicházet ve více kopiích. Proto musí být kontrola doručení paketu realizována programově. Komunikace mezi klientem a serverem iniciuje klient s tím, že veškerá komunikace má tvar požadavek - odpověď. Tudíž odpověď může být chápána jako potvrzení požadavku. V případě, že nepřijde odpověď do časového limitu, je klientem generován nový požadavek s totožným obsahem. Tímto mechanizmem je zaručeno doručení zpráv mezi klientem a serverem.

Špatné pořadí příjmu zpráv či příjem duplikované zprávy serveru ani klientu nevadí, jelikož zprávy obsahují sekvenční číslo. Server při příjmu duplikované zprávy neaktualizuje statistiky e-mailových signatur, pouze generuje novou odpověď, kterou zašle zpět klientu.

Zmíněný časový limit pro příjem odpovědi je průběžně aktualizován, podle naměřených vzorků doby obrátky, anglicky Round Trip Time (RTT). Doba obrátky se měří od odeslání požadavku po příjem odpovědi na daný požadavek. Výpočet RTT a následné stanovení časového limitu, anglicky Time Out (TO), je převzat z protokolu TCP, viz [21]. Jedná se o plovoucí průměr a jeho výpočet je zapsán rov. 5.1. Výpočet TO vyjadřuje rov. 5.2. V rovnicích se používají konstanty s hodnotami: $\alpha = 7/8$, $\beta = 2$, $dolni\_mez = 1$ a $horni\_mez = 60$. Meze vyjadřují počet sekund a definují tak rozsah možného pohybu časového limitu TO. Proměnná RTT_vzorek značí novou naměřenou hodnotu RTT při příjmu odpovědi. Pokud časový limit pro příjem odpovědi bude vyčerpán, použije se jako hodnota RTT vzorku samotná doba časového limitu, tedy TO.

$$RTT_{i+1} = \alpha \cdot RTT_i + (1 - \alpha) \cdot RTT\_vzorek_{i+1} \tag{5.1}$$

$$TO_{i+1} = min(horni\_mez, max(dolni\_mez, \beta \cdot RTT_{i+1})) \tag{5.2}$$

5.3.6 Bezpečnost

Pro zajištění bezpečnosti jsou data v paketech přenášena v šifrované podobě. Při výběru šifrovacího algoritmu se vybíralo mezi algoritmy pracujícími se symetrickým klíčem, z důvodu menších nároků na systémové zdroje. Ze souboru symetrických šifer nakonec bylo přikloněno k použití šifry Advanced Encryption Standard (AES), která je považována za standard v symetrické kryptografii.

AES pracuje s klíčem o délce 128 bitů. Symetrický klíč si server s klientem vymění při přihlašování klienta do systému užitím protokolu HTTPS. Při úspěšném ověření klient obdrží přidělený identifikátor, který je přenášen v každém požadavku na server.

Pro zabezpečení dat bylo rozhodováno, zda se použije streamová šifra Cipher feedback (CFB) nebo bloková šifra Cipher-block chaining (CBC). U streamové šifry není nutno zarovnávat délku přenášených dat do celočíselného násobku velikosti bloku užívaného při šifrování. Velikost bloku šifry AES se rovná 16 bajtům. Nevýhodou je, že zmíněné zarovnání délky dat se provádí automaticky vycpávkou, tudíž délky původních a šifrovaných dat se neshodují. Navíc malá změna v prvním bloku dat se projeví pouze malou změnou šifrovaných dat, jak je zachyceno v tab. 5.1. Z úsporných důvodů je v tabulce vypsáno jen prvních 8 bajtů dat. Za malou změnu lze považovat např. postupně rostoucí sekvenční číslo.

V tab. 5.2 je znázorněno chování blokové šifry při malé změně vstupních dat. Bloková šifra navíc zachovává velikost šifrovaných dat. Z těchto důvodů byla vybrána pro další použití bloková šifra AES CBC, s tím že navržené pakety požadavku a odpovědi jsou navrženy tak, aby délka jejich šifrovaných částí byla bezezbytku dělitelná velikostí bloku použitého při šifrování (16 bajtů).

Data	Hodnoty šifrovaných dat							
abcdabcd	175	179	181	84	95	240	5	99
abc**e**abcd	175	179	181	85	95	240	5	99

Tabulka 5.1: Ukázka výstupu streamové šifry AES-CFB

Data	Hodnoty šifrovaných dat							
abcdabcd	227	159	112	132	27	164	101	108
abc**e**abcd	59	100	191	227	2	157	179	40

Tabulka 5.2: Ukázka výstupu blokové šifry AES-CBC

5.4 Charakteristické signatury

Pro stanovení charakteristických signatur je použit princip popisovaný v kap. 4.5. Pro e-mailovou zprávu, vyhovuje-li diskutovaným požadavkům, je stanovena šestice hodnot tvořící charakteristickou signaturu zprávy. Signatury jsou počítány z předmětu a těla zprávy. Při průchodu e-mailovou zprávou je zjištěn i typ zprávy z pole Content-type.

Jednotlivé položky signatury jsou stanoveny následujícím způsobem, při kterém se použijí rovnice 5.3 až 5.9. Vstupními daty jsou zkoumaná e-mailová zpráva a kombinace

hodnot harmonické složky $i_{harmonicka} = \{1, 2, 3\}$ a velikosti bloku $l_{blok} = \{4, 16\}$. Před výpočtem amplitudy pro každou kombinaci harmonické složky a velikosti bloku dojde ke stanovení normalizované frekvence f_n, koeficientu k a počtu zkoumaných znaků zprávy n. Opakovaným užitím vzorce 5.7 a následným výpočtem dle vzorce 5.8 se stanoví hodnota relativní amplitudy. Na závěr se vzorcem 5.9 určí výsledná amplituda.

$$f_n = \frac{i_{harmonicka}}{2 \cdot l_{blok}} \tag{5.3}$$

$$k = 2 \cdot cos(2 \cdot \pi \cdot f_n) \tag{5.4}$$

$$n = l_{email} - (l_{email} \mod (2 \cdot l_{blok})) \tag{5.5}$$

$$s_{i-2} = 0, \quad s_{i-1} = 0 \tag{5.6}$$

$$s_i = znak_i + (k \cdot s_{i-2}) - s_{i-1}), \qquad \text{pro } i = 1, 2, \ldots n \tag{5.7}$$

$$A_{rel} = s_n^2 + s_{n-1}^2 - k \cdot s_n \cdot s_{n-1} \tag{5.8}$$

$$A = \frac{\sqrt{A_{rel}}}{n} \tag{5.9}$$

Server pro záznamy v databázi se shodným typem zprávy porovnává jednotlivé hodnoty signatury s možnou variabilitou ±1%. Z nalezených záznamů se vybírá jeden s nejvyšší četností, který je aktualizován. Pokud není nalezen žádný, zavádí se do databáze nový záznam s přijatou signaturou.

5.4.1 Význam četnosti

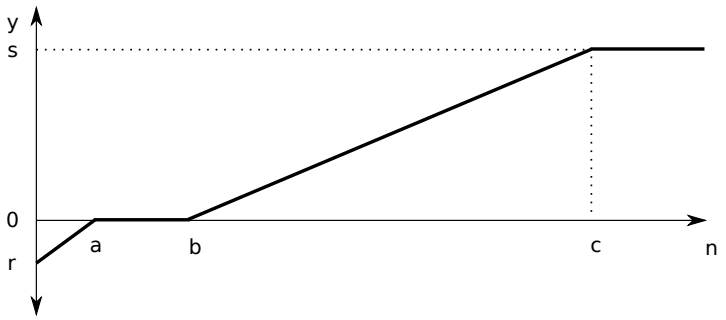
O významu četnosti podobných e-mailových zpráv zatím nebylo nijak polemizováno. Určení hranice, která by třídila e-mailové zprávy podle četnosti na dvě skupiny, kde jedna reprezentuje regulérní poštu a druhá hromadně zasílanou poštu (potenciální spamovou poštu), není přímočaré. Pokud se hodnota hranice zvolí příliš nízká, již při malém množství podobných e-mailových zpráv budou zprávy považovány za spam. S ohledem na velké množství zasílaných e-mailových zpráv v dnešním Internetu je možné, že nezanedbatelné množství těchto zpráv bude prohlášeno za podobné, aniž by ve skutečnosti byly. Hodnota hranice, která rozděluje zprávy na dvě skupiny, by se proto měla pohybovat nejméně v řádech tisíců. U tohoto způsobu je nevýhoda přímého rozdělení zpráv do dvou skupin.

Lepším rozlišovacím mechanismem je použití spamové skóre (dále jen „skóre“), které svojí hodnotou dává najevo šanci, že daná zpráva je spam. V tomto případě lze hodnotu četnosti mapovat na hodnotu dílčího skóre. Získané dílčí skóre se přičte k souhrnnému skóre, na jehož stanovení se podílí více anti-spamových filtrů. Podle nastavených limitů spamového skóre na e-mailovém serveru jsou zprávy označovány jako spamové, příp. rovnou zahazovány.

Pro převod četnosti na hodnotu dílčího skóre je navržena funkce 5.10. Grafická podoba funkce je vystižena na obr. 5.6. Ve funkci jsou použity konstanty: $a = 3$, $b = 10$,

$c = 100$, $r = -0,5$ a $s = 2$. Konstanty jsou zvoleny podle nejlepšího uvážení a s největší pravděpodobností by po nasazení v reálném provozu došlo k jejich úpravě.

$$f(n) = \begin{cases} \frac{r}{a} \cdot n - r & \text{pro } n \text{ menší než } a \\ \frac{s}{c-b} \cdot (min(n, c) - b) & \text{pro } n \text{ větší než } b \end{cases} \tag{5.10}$$



Obrázek 5.6: Funkce pro převod četnosti na spamové skóre

5.5 Řízení výpadků

Klient implementovaný jako knihovna, by měl být nalinkovaný do e-mailového serveru, kde server při zpracovávání každé zprávy zavolá metodu knihovny pro zjištění četnosti výskytu dané zprávy. Proto musí být schopný řádně komunikovat s e-mailovým serverem i v případě výpadku všech serverů implementovaného systému.

Při realizaci klienta bylo definováno celkem pět typů výpadku způsobených různými chybami, viz tab. 5.3.

5.5.1 Časový limit odpovědi

V předešlém textu bylo vysvětleno, jak si klient udržuje aktuální hodnotu doby obrátky RTT, aby mohl pružně reagovat na ztrátu paketů. Je ovšem zapotřebí stanovit maximální hodnotu RTT, při jejíž překročení klient startuje proces přepojení na další server v pořadí, příp. plánuje déle trvající výpadek a následné nové připojení do systému.

V tabulce 5.4 je zachycen růst hodnoty RTT z počáteční hodnoty 1 sekunda. Hodnoty odpovídají situaci, kdy klient nevyužívá paralelního dotazování. Tento růst může nastat např. při výpadku linky. V tabulce je uvedeno prvních šestnáct obrátek. První sloupec tabulky uvádí pořadí obrátky. Poslední sloupec přináší celkový čas od začátku růstu hodnot RTT a TO. Limitní hodnota RTT byla stanovena na hodnotu pěti sekund. Bylo tak vybráno s ohledem na celkový čas (67,23 s), po který klient blokuje e-mailový server, a dobu obrátky. Limitní doba obrátky rovná pěti sekundám poskytuje celkem velký prostor pro pohyb RTT při komunikaci se serverem přes linku se zvýšenou ztrátovostí. Pro nárůst doby obrátky za povolenou mez je zapotřebí čtrnácti neúspěšných obrátek. Pro urychlení detekce nedostupnosti serveru při zahájení komunikace, je nastavena počáteční hodnota RTT na dvě sekundy, což v důsledku sníží celkový čas blokování e-mailového serveru na 50 sekund.

Typ výpadku	Trvání	Popis
BlockedIpError	-	Výpadek způsobený špatným licenčním číslem. Délku výpadku určuje webový server, pomocí kterého se chtěl klient přihlásit do systému. Webový server má přednastavenu hodnotu prodlevy neúspěšného přihlášení na 5 minut.
ConnectionError	20 min.	Při neúspěšném pokusu připojení do systému. Může být vyvolaný i třetím přepojením na další server v pořadí.
LinkStateError	2 sec.	Při překročení limitu pro maximální zpoždění odpovědi ze serveru je použit tento výpadek pro krátké vymezení doby, kdy klient po skončení výpadku začne posílat požadavky na další server v pořadí. Během výpadku je ukončeno zpracování všech zadaných e-mailových zpráv a jsou odevzdány výsledky s nastavenými chybovými kódy.
SocketError	20 min.	Výpadek způsobený chybou soketu. Po ukončení výpadku se klient pokouší soket znovu otevřít.
OtherError	-	Slouží pro dokončení zpracování všech e-mailových zpráv. Výpadek je použit při ukončení programu klienta.

Tabulka 5.3: Typy výpadků klienta

Pokud klient využívá plně paralelizace dotazů, hranice RTT je rovna 5 sekundám a počáteční hodnota RTT rovna 2 sekundám, je celkový čas blokování e-mailového serveru rovný 6,8 sekundám. Pokud by RTT narůstalo po výpadku linky z hodnoty jedné sekundy, celkový čas blokování e-mailového serveru by se rovnal 14,2 sekundám.

Obrátka	RTT [s]	TO [s]	Celkový čas [s]
1.	1,00	2,00	2,00
2.	1,13	2,25	4,25
3.	1,27	2,53	6,78
4.	1,42	2,85	9,63
5.	1,60	3,20	12,83
6.	1,80	3,60	16,44
7.	2,03	4,05	20,49
8.	2,28	4,56	25,05
9.	2,57	5,13	30,18
10.	2,89	5,77	35,96
11.	3,25	6,49	42,45
12.	3,65	7,31	49,76
13.	4,11	8,22	57,98
14.	4,62	9,25	**67,23**
15.	**5,20**	10,40	77,63
16.	5,85	11,70	89,33

Tabulka 5.4: Růst hodnot RTT a TO při výpadku linky

6 Implementace

6.1 Programové prostředky

Před implementací programu serveru a programu (knihovny) klienta se polemizovalo, který programovací jazyk bude použit. Bylo vybíráno mezi programovacím jazykem C, C++ a Java. Po větších praktických zkušenostech s implementací síťových aplikací v jazyce C bylo nakonec přikloněno k implementaci programů diplomové práce v jazyce C++. Touto volbou se chtělo dosáhnout lepšího strukturování aplikace do logických částí. K výběru jazyka C++ přispěl i fakt, že před samotným zahájením implementace programů bylo delší dobu laborováno s vytvářením charakteristických signatur e-mailových zpráv v jazyce C.

Webové rozhraní je implementováno v programovacím jazyce PHP. Není zde použito prostředků HTML ani CSS. Webové rozhraní slouží pouze pro účely bezpečného ověření klienta a výměny počátečních dat, jako je např. symetrický klíč. Jazyk PHP je použit i pro implementaci skriptů, které jsou spouštěny automaticky pomocí cron tabulky a slouží k průběžnému čištění databáze.

Programy byly implementované v prostředí operačního systému GNU/Linux Debian 6. Podrobný popis prostředí i s návodem na jeho zprovoznění je uveden v příloze C.1.

6.2 Knihovny

Při implementaci programů byly použity stávající knihovny.

- **libcurl** Knihovna libcurl [3] umožňuje přenos dat skrze široké spektrum protokolů, např. FTP, FTPS, HTTP a především HTTPS. Podporuje práci s SSL certifikáty, HTTP POST, HTTP GET, apod. Knihovna je portovatelná na mnoho platforem.
- **crypto** Knihovna crypto [1] je součástí souboru nástrojů označených jako OpenSSL, které implementují Secure Sockets Layer (SSL v2/v3) a Transport Layer Security (TLS v1) protokoly. Knihovna crypto se skládá z více knihoven, které implementují jednotlivé algoritmy symetrického, asymetrického šifrování, hašovacích funkcí a dalších funkčností.
- **libxml2** Knihovna pro práci s XML soubory, viz [4].
- **pthreads** Knihovna, celým názvem POSIX threads, tvoří standardizované rozhraní pro práci s vlákny.
- **libmysqlcppconn** Knihovna mysqlcppconn [5] tvoří ovladač propojující databázi MySQL s programovacím jazykem C++.

6.3 Knihovna klienta

Jak již bylo zmíněno, klient je implementován ve formě knihovny, kterou lze použít v dalších programech. Knihovna je zabezpečena pro použití v programech s více vlákny. Jinak řečeno, jedná se o tvz. thread-safe knihovnu.

6.3.1 Rozhraní

Knihovna klienta poskytuje celkem čtyři metody. Metodou *dssdInitialize* se inicializuje samotná knihovna, kdy je jí předán objekt *DssdInitObject*, který obsahuje lokální adresu a port pro navázání soketu, licenční číslo, úroveň logování a výstupní logovací soubor. K řádnému ukončení činnosti knihovny serveru slouží metoda *dssdTerminate*. E-mailový server má možnost zadávat požadavky na zpracování e-mailové zprávy dvěma způsoby, synchronně (*processMessage*) či asynchronně (*processMessageCallback*). U synchronního volání dochází k blokování volajícího vlákna až do doby, kdy je možno vrátit ohodnocení zkoumané e-mailové zprávy. U asynchronního volání volající vlákno není blokováno a výsledné ohodnocení zprávy je předáno zpětným voláním definované metody (callback). Synchronní a asynchronní volání lze libovolně kombinovat.

6.3.2 Logický rozklad

Klient se skládá z několika logických celků. Diagram tříd je umístěn v příloze B.2. Veškeré konstanty užívané knihovnou jsou definované v hlavičkovém souboru *config.h*.

Jádrem knihovny je třída *Client*. Ta je navržena dle návrhového vzoru singleton. Instance třídy je vytvořena při inicializaci knihovny, kdy se vytvoří nové vlákno, které realizuje veškeré činnosti klienta, kromě příjmu odpovědi ze serveru. Po většinu doby vlákno čeká na příchozí události, tj požadavek ke zpracování e-mailové zprávy, příp. odpověď ze serveru. Funkčnost výkonného vlákna klienta lze zachytit vývojovým diagramem na obr. B.4. Ve vývojovém diagramu jsou šedým zvýrazněním označeny činnosti vlákna, které nejsou vykonávány pod uzamčeným výhradním zámkem. Tento zámek e-mailový server zamyká v metodě *processMessage* či *processMessageCallback*, díky čemuž je zabezpečena korektní kontrola výpadku klienta a bezchybné předání nové zprávy ke zpracování.

Vlákno je vytvořeno voláním příslušných funkcí knihovny *pthread*. Čekání na příchozí události se realizuje časovaným uzamčením nad podmínkovou proměnnou, metoda *pthread_cond_timedwait*. Výhodou tohoto způsobu čekání na událost je, že se čekání ukončí po vypršení časového limitu. Této vymoženosti je využito pro realizaci časových limitů, jak pro ukončení výpadků serveru, tak pro opakované odeslání požadavku na server.

Klient uchovává veškerá data, potřebná ke svému chodu, v instanci třídy *Storage*, která tak tvoří tzv. kontext knihovny. Z velkého množství proměnných obsažených v kontextu můžou být jmenovány důležitější proměnné jako např. symetrický klíč, identifikátor klienta, sekvenční číslo, seznam serverů, seznam zpracovávaných e-mailových zpráv, příznaky výpadku klienta i čas konce výpadku, v případě že se klient zrovna v nějakém nachází.

Modul pro přihlášení klienta do systému je tvořen dvojicí tříd *Https* a *XmlConnection*. První zmiňovaná třída využívá knihovny *libcurl* pro přihlášení se do systému přes protokol HTTPS a získání inicializačních dat ve formátu XML, viz výpis 6.1. Získaná data jsou ve třídě *XMLConnection* rozebrána a v případě pozitivní odpovědi webového serveru, který je původcem přijatých dat, je nastaven kontext knihovny, která poté může zpracovávat e-mailové zprávy.

Pro vytváření charakteristických signatur e-mailových zpráv slouží třída *Signature*. Před vytvořením charakteristické signatury se načte prvních 20 kB dané e-mailová zprávy do paměti. Vyhledá se typ zprávy, který je určený položkou hlavičky *Content-type*. Vyhledá se začátek těla zprávy, před který se zkopíruje předmět zprávy. Na závěr se z před-

mětu a těla zprávy stanoví Goertzelovým algoritmem šest hodnot charakteristické signatury, viz kap. 5.4.

Charakteristické signatury s typem e-mailu a dalšími režijními daty jsou uloženy do struktury *udp_request*, která symbolizuje paket požadavku, zasílaného na server. Struktura pro manipulaci s jednotlivými položkami paketu nabízí metody, které obsahují převod z hostitelské reprezentace dat na síťovou a naopak. Před odesláním jsou všechna data kromě identifikátoru zašifrována. O šifrování a dešifrování paketů se stará třída *Secure*.

Třída *Udp* nabízí metodu pro odeslání požadavku na server. Při inicializaci knihovny klienta se otevírá soket, přes který bude realizována komunikace mezi klientem a serverem. Oproti ostatním třídám, třída *Udp* obsahuje vlákno, které se stará o příjem paketů s odpověďmi serveru. Přijatý paket je uložen do fronty příchozích odpovědí pouze tehdy, když jeho délka přesně odpovídá délce paketu pro odpověď a když případné vložení do fronty nezpůsobí překročení limitu na maximální délku fronty. Do fronty je vkládán paket tak jak je. O jeho dešifrování a následné zpracování se stará výkonné vlákno klienta, jinak řečeno vlákno singletonu *Client*.

Pro jednoduchou manipulaci s časem, konkrétně se strukturou *timespec*, je vytvořena třída *TimeTools*. Struktura *timespec* je použita v metodě *pthread_cond_timedwait* pro časované uzamčení nad podmínkovou proměnou.

Třída *Log* poskytuje metodu pro vypisování kontrolních výpisů na standardní výstup či do souboru specifikovaného při inicializaci knihovny. Úroveň logování lze ovlivnit hodnotou zadané masky. Jednotlivé úrovně logování jsou vztaženy k logickým celkům programu. Výpis úrovní lze libovolně kombinovat sečtením jejich hodnot masky. Speciální úroveň s názvem *LogAll* způsobí výpis všech kontrolních záznamů. Seznam úrovní kontrolních výpisů lze nalézt i s jejich číselnými hodnotami v příloze D.1. V následujícím ukázce je uveden příklad zápisu kontrolního výpisu v kódu a jeho výstupní podoba:

```
Log::write(LogNetwork | LogError,
        "cannot bind socket to address %s:%d\n", address, port);

2012-04-24 21:59:31 [network ERROR] cannot bind socket to address
172.16.119.1:20011
```

6.3.3 Návratové kódy

Při volání knihovních funkcí klienta lze obdržet návratové kódy obsažené v tab. 6.1. Stejné návratové kódy jsou používané při předání výsledků zpětným voláním (voláním callback metody).

6.4 Klient

Pro ověření funkčnosti knihovny byl vytvořen program, který vytvořenou knihovnu klienta používá. Program se skládá z jediného souboru *main.c*, ve kterém se zpracují vstupní parametry, jimiž se ovlivňuje chování programu. Klient inicializuje knihovnu a vytváří požadavky na zpracování e-mailových zpráv. Následně vypisuje názvem souboru, četnost zprávy a stanovené spamové skóre e-mailové zprávy na standardní výstup. E-mailové zprávy, které se mají zpracovat, se zadávají při spouštění programu buď přímo výčtem

Návratový kódu	Hodnota	Poznámky
NoError	0	Bez chyby.
AlreadyInitialized	1	Knihovna již byla inicializována.
BindSocketError	2	Nepodařilo se otevřít soket pro komunikaci se serverem.
CreateThreadError	3	Chyba při vytváření vláken.
LogFileOpenError	4	Knihovna nemůže otevřít log soubor.
NotInitialized	5	Knihovna nebyla inicializována.
SystemOutage	6	Klient se nachází ve stavu výpadku.
EmailBodyNotFound	11	Nebylo nalezeno tělo zprávy.
EmailFileNotFound	12	Zadaný soubor zprávy neexistuje.
EmailFileReadError	13	Chyba při čtení souboru zprávy.
EmailIsTooShort	14	Tělo zprávy je příliš krátké pro zpracování.
EmailProcessingFail	15	Chyba při zpracovávání zprávy.
EmailsCountLimitAchieved	16	Dovršen maximální limit současně zpracovávaných zpráv.

Tabulka 6.1: Návratové kódy knihovny klienta

jejich souborů nebo zadáním složky, ve které jsou zprávy obsaženy. Klient umožňuje rekurzivní průchod složkou.

6.5 Server

Program serveru se dělí do logických celků reprezentovanými třídami, jež jsou obsaženy v diagramu tříd, viz příloha B.3. Obdobně, jako tomu bylo u klienta, jsou serverem užívané konstanty definované v hlavičkovém konfiguračním souboru *config.h*.

Po spuštění programu serveru je po zpracování vstupních parametrů zkontrolován přístup do databáze. Po úspěšné kontrole jsou postupně vytvořeny objekty tříd *RequestQueue*, *Udp*, *Monitor* a *ThreadPool*.

Třída *RequestQueue* spravuje frontu příchozích požadavků na server. Do fronty se ukládají přijaté požadavky ve stejné podobě, jaké jsou obdrženy přijímacím vláknem třídy *Udp* ze soketu. Při odebírání požadavků z fronty, jsou obslužná (výkonná) vlákna serveru uzamykána nad podmínkovou proměnnou v případě, že je fronta prázdná. Maximální délka fronty pro uložení přijatých paketů je definována konstantou *MAX_QUEUE_LENGTH* v konfiguračním souboru. Stanovení hodnoty této konstanty vychází z maximální doby obsloužení požadavku a z naměřených dat při testování serveru. Horní hranice doby obsluhy vyplývá ze zvolené mezní hodnoty RTT na klientu, kdy klient při jejím překročení začne zasílat požadavky na jiný server. V kap. 5.5.1 byla hranice RTT stanovena na hodnotu pěti sekund. Hodnota maximální doby obsluhy požadavku musí být menší než zvolená hranice RTT. To vyplývá z následující úvahy. Pokud by server odpovídal na všechny požadavky se zpožděním pěti sekund, tak hodnota RTT jeho klientů v konečné době dosáhne hranice též pěti sekund. Když se k tomu přidá i zpoždění linky, bude docházet k neustá-

lému přepojování klientů mezi servery. Proto se jako maximální doba obsluhy požadavku použila hodnota tří sekund. Z výsledků stresového testování uvedených v kap. 7.2.6 lze vyčíst průměrnou délku fronty v okamžiku, kdy se doba obsluhy pohybovala kolem tří sekund. Průměrná délka fronty se rovnala 103 požadavkům, a tak maximální délka fronty pro příjem požadavků byla stanovena na hodnotu 110 požadavků.

Třída *Udp* obsahuje vlákno pro příjem požadavků ze soketu a poskytuje metody pro odesílání odpovědí jednotlivým klientům. Přijatý požadavek ze soketu se uloží do struktury *udp_encoded_request*, která mimo samotného požadavku obsahuje délku požadavku, čas přijetí a adresu odesílatele požadavku. Odpověď na požadavek se zasílá na adresu odesílatele obdrženou při příjmu paketu. Nepoužívá se adresa vedená v databázi, aby se neodepřela možnost použití systému klienty, kteří jsou zatíženi překladem[1] IP adres a především čísel portů.

Třída *Monitor* slouží pro periodické měření vytížení serveru a aktualizaci naměřených hodnot v databázi. Pro monitorování se spouští samostatné vlákno, které se aktivuje v definovaných časových intervalech a provádí kýženou aktualizaci, kde se do databáze ukládá momentální procentuální vytížení fronty přijatých požadavků. Monitor je obohacen o funkci průběžné kalkulace a zaznamenávání dalších výkonnostních parametrů, jako např. průměrná doba odpovědi na požadavek. Způsob povolení rozšiřující funkce monitoru je popsán v kap. C.3. Naměřené výkonnostní parametry se vypisují na standardní výstup *stdout* ve formátu: *„datum a čas;počet zpracovaných požadavků;průměrná délka fronty;průměrné vytížení CPU;průměrná doba obsluhy“*. Pro měření výkonnostních parametrů je zavedena třída *Measuring*, která obsahuje statické proměnné pro uchování měřených hodnot a poskytuje metody pro manipulaci s nimi.

O obsluhu a zpracování požadavků se stará soubor dvou vláken, které definuje třída *ServerThread*. Vlákna jsou spravována třídou *ThreadPool*, jež poskytuje metody pro spuštění vláken a čekání na ukončení běhu vláken. Běh vláken se ukončí přes frontu požadavků, která při výběru požadavku z fronty vrátí vláknu pointer rovný nule, načež vlákno reaguje svým ukončením. Vlákna instancí třídy *ServerThread*, při obdržení požadavku z fronty, identifikují klienta podle prvního pole identifikátor obsaženého v paketu požadavku. Pokud je klient obsažen v databázi a nevypršel mu časový limit spojení, dešifruje se zbytek dat užitím symetrického klíče klienta. Následně se ověří kontrolní součet, a sekvenční číslo. Kontrola sekvenčního čísla se provede v databázové transakci, kdy se zjistí aktuální stav sekvenčního čísla a bitové mapy klienta, provede se kontrola a příp. oprava údajů. Na závěr se upravené údaje uloží zpět do databáze. Transakce lze zachytit pseudokódem uvedeným na obr. 6.1.

```
získej data klienta z databáze a uzamkni záznam
pokud byl klient nalezen
     pokud je sekvenční číslo v povolených mezích
          uprav data klienta
          ulož změny do databáze
          commit
     jinak
          rollback
jinak
     rollback
```

Obrázek 6.1: Pseudokód transakce pro úpravu dat klienta v databázi

[1] Network Address Translation (NAT)

Server šifruje a dešifruje data paketů užitím metod instance třídy *Secure*. Na rozdíl od klientské aplikace, se inicializuje šifra odpovídajícím symetrickým klíčem před každým dešifrováním dat požadavku a následně se uvolňují prostředky po každém šifrování dat odpovědi.

V předchozím textu se uváděla zmínka o periodické aktualizaci databáze v monitorovacím vlákně. Pro práci s databází se používá konektor, který je tvořen instancí třídy *MysqlConnector*, jež ke své činnosti využívá knihovnu *libmysqlcppconn*. Před použitím konektoru musí dojít k prvotnímu nastavení konektoru, kde se naváže spojení s databází a kde se nastaví použitý mód pro automatické potvrzování transakcí, tzv. „autocommit“. Monitorovací vlákno užívá konektoru s automatickým potvrzováním. Vlákna instancí třídy *ServerThread* si pro komunikaci s databází navazují hned dva konektory. První konektor s povoleným autocommit módem slouží k aktualizaci e-mailových signatur. Druhý konektor má autocommit mód vypnutý, a to z důvodu aktualizace záznamu o klientu, kde se zprvu vybere aktuální záznam o klientu z databáze, provedou se kontroly sekvenčního čísla, nastaví se bitová mapa, upraví se hodnota sekvenčního čísla a uloží se záznam zpět do databáze. Záznam se neaktualizuje v případě, že sekvenční číslo přijatého požadavku nespadá do povoleného rozmezí.

6.6 Webový server

Webový server je psaný ve skriptovacím jazyce PHP. Slouží pouze k ověření klienta, kterému vrací data ve formátu XML, viz výpis 6.1.

Aplikace webového serveru je rozdělena do tří vrstev, kde první vrstva tvoří datový model a druhá vrstva definuje řídící logiku. Třetí vrstva, jinak známá jako prezentační ve spojitosti s architekturou Model-View-Controller, nebyla pro účely diplomové práce téměř využita. Nutno podotknout, že základ webové aplikace byl použit ze semestrální práce předmětu KIV/ASWI[2]

Kromě rozdělení aplikace do vrstev se užitím tříd realizoval rozpad aplikace na menší logické celky. V následujícím textu budou vyzdviženy jen některé třídy. Důležitou sadu tříd tvoří třídy sloužící k objektově relačnímu mapování dat databáze. Jsou jimi třídy *BlockedIp*, *Client*, *Email*, *Licence*, *Server*. Podstatnou roli sehrává třída *Configuration*, jež se stará o načítání konfiguračních dat ze souboru *config/configuration.xml*.

O obsluhu požadavků směřujících na webový server se stará třída *Dispatcher*, jejíž instance je spouštěna s každým příchozím požadavkem na server. Podle typu požadavku předává řízení příslušné třídě, která se stará o obsluhu daného typu požadavku. Na webový server je zasílán jediný typ požadavku, reprezentující přihlášení klienta do systému, který zpracovává třída *ConnectAction*. Vstupními parametry přihlášení jsou sériové číslo klienta a číslo verze podporovaného protokolu. Odpovědí na přihlášení klienta jsou data formátovaná jazykem XML, viz výpis 6.1. Položka odpovědi *status_code* udává, zda-li přihlášení proběhlo úspěšně (hodnota rovná nule). Ostatní předávané parametry slouží k počátečnímu nastavení klienta. Jelikož je přihlášení do systému omezeno na určitou dobu, po které musí klient opakovat přihlášení, je mu položkou *session_durability* oznámena doba trvanlivosti přihlášení. Není použit konkrétní čas, nýbrž časový úsek, díky čemuž není vyžadována časová synchronizace serveru s klientem.

[2]KIV/ASWI, projekt One Click Release, školní rok 2010/2011, autoři: Daniel Gruber, Lubomír Jurečka, Martin Petrák, Jiří Praus

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
 <status_code>0</status_code>
 <status>OK</status>
 <identifier>475835802</identifier>
 <sequence_number>3065</sequence_number>
 <window_size>10</window_size>
 <session_durability>86100</session_durability>
 <symmetric_key>21deb2579162168f</symmetric_key>
 <servers count="3">
  <server ip="172.16.119.128" port="30001"/>
  <server ip="172.16.119.128" port="30002"/>
  <server ip="172.16.119.128" port="30003"/>
 </servers>
</response>
```

Výpis 6.1: Data ve formátu XML získaná po úspěšném přihlášení do systému

Aplikace webového serveru obsahuje skript *CleanUp*, který je spouštěn automatický spouštěn plánovačem úloh (démon cron) a slouží pro mazání zastaralých záznamů databáze. Hodnoty definující pomyslné stáří záznamů jsou specifikované v konfiguračním souboru. Konkrétně se jedná o záznamy klientů a e-mailových signatur. E-mailové signatury s větší četností, neboli spamovým skórem, jsou drženy v databázi delší dobu.

6.7 Databáze

Jako databázový server se použil server MySQL, který byl vybrán s ohledem na snadnou konfiguraci a použití ve spojení s webovým server Apache a skriptovacím jazykem PHP.

Model databáze je umístěn v příloze B.1. Databáze obsahuje pět tabulek.

Tabulka *licence* obsahuje seznam všech aktivních licenčních čísel e-mailového serveru Kerio Connect. Mimo položek definující rozsah licence, které se v současné době nijak nepoužívají, je v tabulce obsažena položka *rank* definující důvěryhodnost klienta používající dané licenčního číslo.

V tabulce *client* jsou drženy záznamy o všech přihlášených klientech do systému. Záznam o klientu obsahuje např. identifikační číslo klienta, důvěryhodnost klienta (*client_rank*), symetrický klíč a termín expirace přihlášení. Záznam o klientu dále obsahuje položky nutné ke kontrole sekvenčního čísla a kontrole duplicity dotazu: velikost okna, bitovou mapu a sekvenční číslo. Ostatní položky tabulky mají informativní charakter.

Tabulka *email* slouží pro uchování e-mailových signatur. Skládá se z typu e-mailové zprávy a prostoru pro uchování samotné signatury, která se skládá ze šesti hodnot. Dalšími položkami tabulky jsou četnost výskytu *count*, spamové skóre *spam_rank* a doby posledního výskytu e-mailové signatury.

Posledními tabulkami obsaženými v databázi jsou tabulky *blocked_ip* a *server*. První jmenovaná tabulka slouží k uchování blokovaných IP adres, ze kterých došlo k neúspěšnému pokusu přihlášení do systému, jež byl způsobený neznámým licenčním číslem. Položka *expiration* vymezuje dobu, po kterou nesmí dojít k opakovanému pokus o přihlášení. Poslední položka má pouze informativní charakter a udává počet pokusů o přihlášení vedených z dané blokované IP adresy.

Tabulka *server* je určena pro uchování seznamu běžících serverů a jejich vytížení.

7 Ověření funkčnosti

Kapitola si klade za cíl ověřit praktické použití implementovaného systému a poukázat na jeho slabé a silné stránky. V první části bude ověřena samotná funkčnost systému, kde bude zkoumáno chování klienta a serveru i za ztížených podmínek. V druhé části se budou sledovat výkonnostní parametry systému. Třetí část je věnována z kvalitě navržené tvorby signatury.

7.1 Funkčnost systému

Klient je ve skutečnosti pouze knihovna, určená pro použití v produktu Kerio Connect, byl proto vytvořen program *dssd_client* pro simulaci cílové aplikace. Tento program využívá vytvořenou knihovnu pro stanovení spamového skóre jednotlivých e-mailových zpráv.

Knihovna z podstaty svého určení a způsobu užívání nesmí nijak ovlivnit korektní běh aplikace, která ji využívá. Při neobvyklých událostech, jako jsou například pád serveru, nedostupnost serveru či ztráta paketu, se musí klient uvést do neaktivního stavu a na vnější požadavky od aplikace reagovat patřičnými chybovými výstupy.

Při kontrole funkčnosti byly proto kontrolovány chybové výstupy knihovny klienta za současného pozorování výstupu logů. Funkčnost systému byla ověřena testovými případy z tab. 7.1 a 7.2, kde všechny testové případy byly funkční.

Testování funkčnosti probíhalo na notebooku, jehož specifikace je vypsána v příloze F.2. Program klienta byl spuštěn přímo v operačním systému daného zařízení, programy serverů byly spuštěny ve virtuálním stroji, viz příloha C.2. Virtuální stoj programu VMware Player nabízí nastavení procentuální ztrátovosti všech síťových rozhraní v obou směrech separátně, čehož bylo hojně využito pro simulaci ztrát paketů.

Při testování byl využit vzorek e-mailové komunikace Enron Email Dataset [18] z roku 2009, který čítá přibližně půl milionu e-mailových zpráv. Vzorek je umístěn na přiloženém médiu ve složce *email_corpus/*.

Testový případ	Poznámky
Blokující volání knihovní metody klienta	Hlavní vlákno programu zablokováno, až do doby, kdy je e-mailová zpráva zpracována. Je nastaveno ohodnocení e-mailové zprávy, případně je vrácen chybový kód.
Neblokující volání knihovní metody klienta	Hlavní vlákno není blokováno. V případě chyby je okamžitě vrácen chybový kód. Po zpracování e-mailové zprávy je předán výsledek při úspěchu, případně chybový kód při neúspěchu.
Odpojení linky od virtuálního stroje	Dochází k postupnému nárůstu časového kvanta, neboli „timeoutu“, pro opakování odeslání požadavku na server. Při překročení definované hranice se klient po krátkém setrvání ve stavu výpadku přepne na jiný server.

Tabulka 7.1: Testové případy

Testový případ	Poznámky
Odpojení linky od virtuálního stroje a opětovné připojení	Za podmínky, kdy timeout nepřekročil definovanou hranici se pokračuje v přenosu dat.
Ukončení serveru v průběhu zpracování požadavků od klienta	Klient se chová stejně, jako při odpojení linky. Po opětovném startu serveru se pokračuje v přenosu dat.
Odebrání IP adresy	Pokud byla specifikována IP adresa při startu klienta, detekuje se chyba při odeslání dat a klient se přepne do stavu výpadku soketu. V případě, kdy nebyla specifikovaná IP adresa, se událost projeví jako výpadek linky a při návratu IP adresy se pokračuje v přenosu dat.
Deaktivace síťového zařízení	Událost se projeví stejně jako výpadek linky. Při aktivaci síťového rozhraní se pokračuje v přenosu dat.
Nedostupný webový server	Po třech neúspěšných pokusech o připojení se klient na definovaný čas přepne do stavu výpadku. Po vypršení času určeného pro výpadek klient opakuje pokus o připojení do systému.
Ztrátovost linky 5% v obou směrech s paralelizací 10-ti požadavků	Všechny požadované e-mailové zprávy, o celkovém počtu 1000 zpráv, byly vyřízeny za odeslání 1108 požadavků na server v celkové době 129 sekund. Průměrná hodnota RTT při tom činila 1,02289 sekund.
Ztrátovost linky 20% v obou směrech s paralelizací 10-ti požadavků	Všechny požadované e-mailové zprávy, o celkovém počtu 1000 zpráv, byly vyřízeny za odeslání 1572 požadavků na server v celkové době 588 sekund. Průměrná hodnota RTT přitom činila 1,27540 sekund.
Ztrátovost linky 5% v obou směrech bez paralelizace	Všechny požadované e-mailové zprávy, o celkovém počtu 1000 zpráv, byly vyřízeny za odeslání 1101 požadavků na server v celkové době 202 sekund. Průměrná hodnota RTT přitom činila 1,01523 sekund.
Ztrátovost linky 20% v obou směrech bez paralelizace	Všechny požadované e-mailové zprávy, o celkovém počtu 1000 zpráv, byly vyřízeny za odeslání 1593 požadavků na server v celkové době 1364 sekund. Průměrná hodnota RTT přitom činila 1,16512 sekund.

Tabulka 7.2: Testové případy II

7.1.1 Zhodnocení

Navržený systém se chová podle předpokladů, které byly rozebrány v dřívějších kapitolách. Při testování funkčnosti se pracovalo s různými úrovněmi ztrátovosti paketů a byly vytvářeny nahodilé události jako výpadky serverů, linky či webového serveru.

7.2 Zátěžové testování

Zátěžové testování [25] je z obecného pohledu chápáno jako proces, při kterém se vytvoří určitá zátěž zaměstnávající server za současného monitorování jeho odezvy a dalších měřitelných parametrů.

7.2.1 Typy testů

Zátěžové testování lze dělit do skupin podle způsobu a cíle testování. Skupiny tvoří testování na pozadí, stresové testování a výkonnostní testování.

Testováním na pozadí je chápáno jako klasické funkční testování systému, který je ovšem současně zaměstnán zpracováním reálné zátěže.

V průběhu stresového testování je zátěž zvýšena do takových úrovní, aby se testovaný systém ocitl na pokraji svých možností. Cílem tohoto způsobu testování je odhalení nedostatků systému, které se projevují nebo jsou lépe patrné, když je systém vytížen na samý okraj svých schopností. Stresovým testováním se získá přehled o výkonových a kapacitních mezích systému.

Výkonnostní testování má dokázat, že systém splňuje výkonnostní požadavky nebo cíle v závislosti na dané zátěži. Při výkonnostním testování se obvykle měří propustnost, doba odezvy a spotřeba zdrojů, například využití operační paměti (RAM), diskové paměti a procesorových jednotek (CPU).

7.2.2 Faktory ovlivňující testování

Měřené hodnoty mohou být při testování systému ovlivněny mnoha faktory. K těm důležitějším patří např. možnost paralelního dotazování, počet záznamů v databázi, počet klientů zatěžující servery a počet serverů samotných. Jmenované faktory tvoří skupinu faktorů, se kterými se bude při ověřování funkčnosti určitým způsobem laborovat.

7.2.3 Způsob měření

Server měří veškeré výkonnostní parametry v průběhu zpracování požadavků, pokud je přeložen do módu s rozšířeným monitorováním, viz příloha C.3. Měřenými výkonnostními parametry jsou propustnost, doba zpracování požadavku, počet požadavků ve frontě a vytížení CPU. Délka fronty je snímána při každém odebrání požadavku z fronty.

Pro účely testování jsou vytvořeny skripty pro automatizované spouštění klientů. Detailní popis skriptů je uveden v příloze C.4.1. Skripty vypisují při každém spuštění klienta datum a čas, čímž je umožněno sledovat výkonnostní parametry serverů v souvislosti s počtem běžících klientů. Pro korektních korespondenci serverem naměřených hodnot a časů spuštění jednotlivých klientů je zapotřebí synchronizovat čas všech počítačů, kteří se na testování podílejí.

7.2.4 Metodika testování

V průběhu testu jsou se zadaným časovým rozestupem spouštěny programy klientů, buď jednotlivě, nebo po určitých dávkách.

Pro účely zátěžového testování byl klient přeložen do speciálního módu, viz příloha C.3, ve kterém jsou signatury e-mailových zpráv generovány náhodně. Klient tak není nucen zpracovávat jednotlivé soubory. Ušetří se zdroje při vytváření většího počtu klientů a zároveň nebudou jednotliví klienti generovat shodnou zátěž. Při generování náhodných signatur se tak lépe simuluje reálná zátěž. Předpokladem je veliká rozmanitost signatur, s čímž souvisí nutnost zavádění těchto signatur do databáze.

Při zátěžovém testování byly měřeny parametry popsané v tab. 7.3.

Parametr	Jednotky	Popis parametru
Propustnost	pož./s	Počet požadavků zpracovaných za sekundu.
Doba odezvy	s	Doba od přijmu požadavku do fronty po jeho zpracování a odeslání odpovědi.
Počet požadavků ve frontě		Počet požadavků umístěných ve frontě přijatých požadavků.
Vytížení CPU	%	Průměrné využití procesoru počítače.

Tabulka 7.3: Měřené parametry při zátěžovém testování

7.2.5 Testovací prostředí

Pro zátěžové testování byly použity dva virtuální stroje, viz F.2. První virtuální stoj určený pro server byl spuštěný na ESX serveru. Druhý virtuální stroj určený pro simulaci klientů běžel na vlastním PC. Jelikož byly na ESX serveru zprovozněny i další virtuální stroje, které výrazně ovlivňovaly naměřené výsledky, bylo pro zátěžové testování využito i testovací prostředí sestávající se z notebooku, na kterém se spouštěli klienti, a PC, jež bylo určeno pro serverovou část. Na daných zařízeních běžely programy přímo v operačních systémech (nepoužila se virtualizace). Specifikace druhého testovacího prostředí je obsažena v příloze F.1. Testování probíhalo v obou případech na lokálním segmentu sítě.

7.2.6 Výsledky

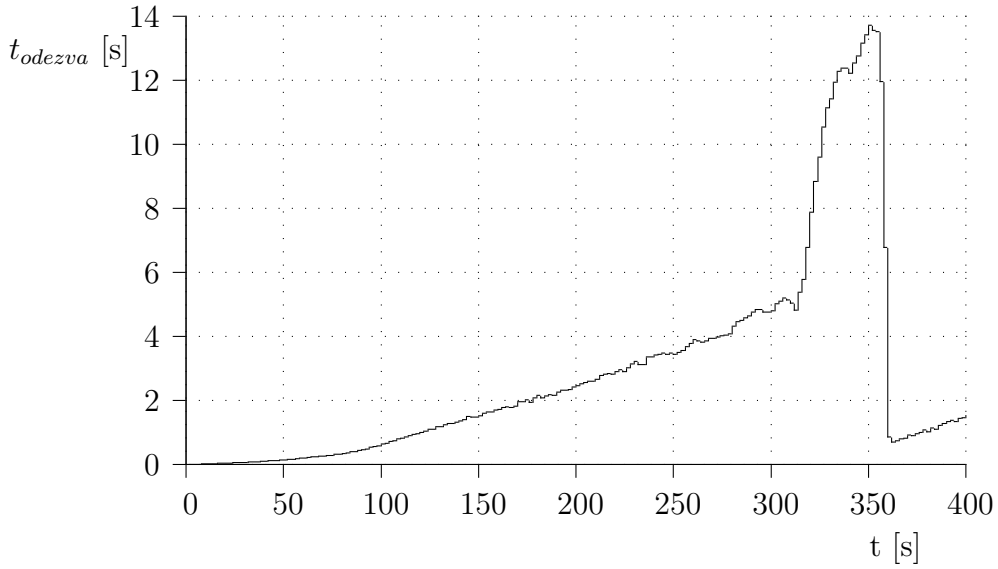
Kompletní výsledky níže uvedených měření jsou umístěny na přiloženém médiu ve složce *test_results/*.

Stresové testy

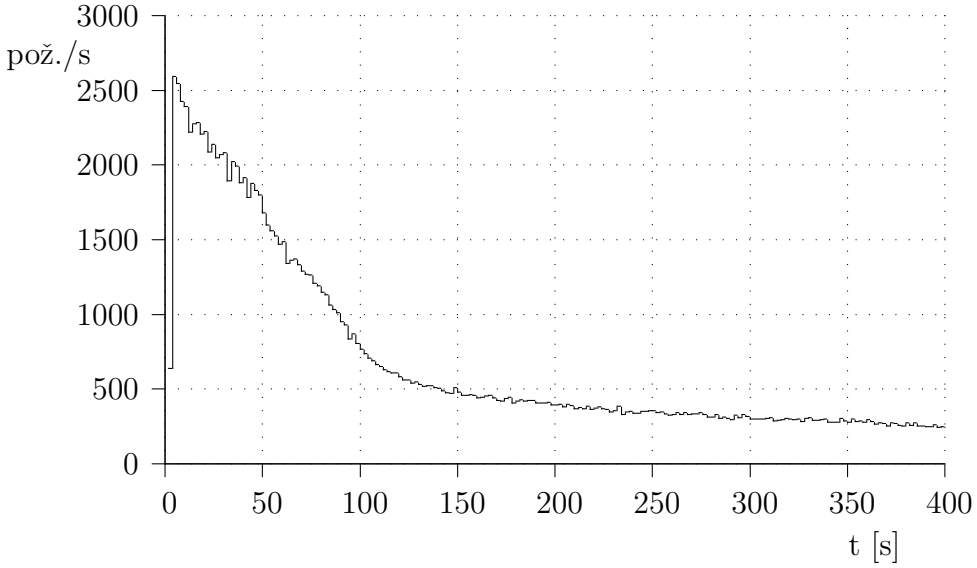
Pro zjištění výkonových hranic systému byla vytvořena zátěž v podobě 300 klientů, jež se spouštěli po dvou ve 30 sekundových rozestupech. Klienti využívali paralelizace dotazů. Na obr. 7.1 jsou zobrazeny naměřené výsledky, kde v části (a) je vynesena průměrná doba odezvy serveru. Je patrné, že za hranicí 150 sekund od začátku testu došlo k postupnému přepojení klientů na další server v pořadí. Jelikož byl v systému spuštěn jediný server, klienti tak své požadavky směřovali opět na daný server. To zapříčinilo nárůst fronty požadavků a tím se zvýšila i doba odezvy. Klienti se tímto způsobem přepojili celkem dvakrát. Až zhruba po 28 sekundách došlo u klientů k výpadku typu *ConnectionError*. Výpadek je zachycen náhlým poklesem doby odezvy na hodnotu 0,7 sekundy.

Maximální doba odpovědi trvalejšího charakteru, kterou klient je schopen akceptovat, je pět sekund. Aby se klient nepohyboval na hraně, kdy by mu neustále hrozil výpadek, musí být doba odezvy serveru menší. Jako vhodná maximální doba odezvy se jeví hodnota tří sekund. Z toho vyplyne i vytvoření zátěže pro výkonnostní testování. Při hodnotě doby odezvy rovné třem sekundám zatěžovalo server celkem 118 klientů a průměrná délka fronty dosahovala hodnoty 1116 požadavků.

Na obr. 7.1 v části (b) je zobrazen počet zpracovaných požadavků. Je patrný zpočátku prudší pokles, který je způsobený postupným plněním databáze, která byla na začátku testu prázdná. Postupem času se pokles zpomalil. V době kdy server zpracovával požadavky s odezvou rovnou třem sekundám, bylo obslouženo 362 požadavků (měřeno pro časový úsek rovný jedné sekundě) a vytížení CPU se rovnalo 24%.



(a) průměrná doba odezvy



(b) propustnost

Obrázek 7.1: Grafické znázornění výsledků stresového testování I

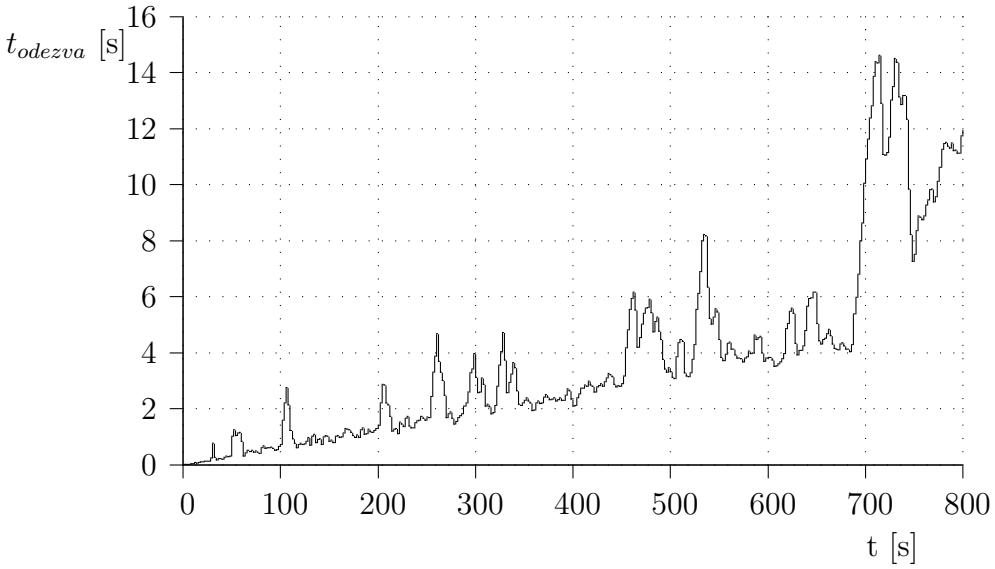
Na daném prostředí se provedl stresový test znovu, tentokrát s naplněnou databází, jež obsahovala 208 277 záznamů e-mailových signatur. Grafy pro tento test zde nebudou zobrazeny, jelikož se velice podobají výše uvedeným grafům. Na rozdíl od předchozího měření se počty zpracovaných požadavků za jednu sekundu téměř ustálily na hodnotě 67. Při hodnotě odezvy rovné třem sekundám komunikovalo se serverem 21 klientů a fronta obsahovala 200 nevyřízených požadavků. Tyto hodnoty budou uvažovány při výkonnostním testování.

Výsledky stresového testu spuštěného na virtuálních strojích jsou zobrazeny v podobě grafů na obr. 7.2. Při testu bylo spuštěno postupně po jednom 200 klientů (bez paralelního dotazování) s časovými rozestupy dvě sekundy. V databázi bylo před testem 280 246 e-mailových signatur. Průběhy grafů jsou ovlivněny činností ostatních virtuálních strojů na ESX serveru. Při odezvě rovné třem sekundám server zpracovával 34 požadavků za sekundu, fronta obsahovala 103 nevyřízených požadavků, CPU bylo vytíženo na 40% a server zatěžovalo 118 klientů.

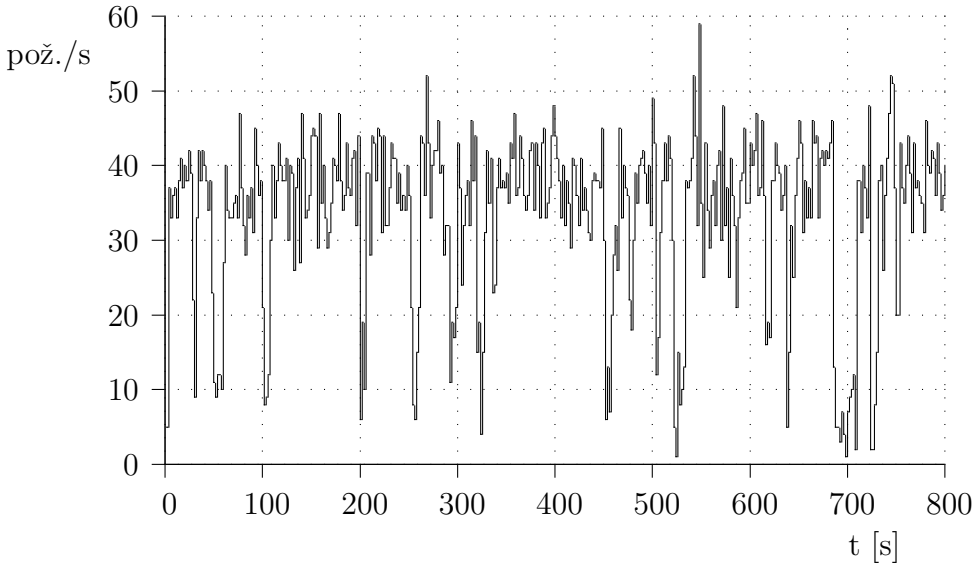
Výkonnostní testy

První výkonnostní test byl spuštěn v prostředí dvou počítačů (notebook + PC) bez využití virtualizace. Podle dříve stanovených parametrů se testu zúčastnilo 21 klientů s paralelním dotazováním. Maximální délka fronty na serveru byla nastavena na 210 požadavků. V databázi před testem bylo uloženo 227 113 záznamů e-mailových signatur. Po skončení testu čítala databáze celkem 809 495 e-mailových signatur. Jelikož byly počáteční parametry zvoleny nevhodně, byl na začátku testu ukončen běh celkem osmi klientů. Z grafů na obr. 7.3 lze vypozorovat neustálý pokles výkonnosti serveru, způsobený nárůstem databáze. Po korekci počtu běžících klientů byl spuštěn druhý server. Po překročení povolené doby RTT na některých klientech došlo k výpadku typu *ConnectionError*, jež trval 20 sekund. Tato událost je zachycena výrazným zlepšením doby odezvy v čase 8 792 sekund. Po skončení výpadku je patrné využití i druhého serveru. Před koncem testu je pozorovatelný ještě jeden výpadek typu *ConnectionError*. Před samotným výpadek došlo k přepnutí některých klientů z prvního serveru na druhý. V části (b) obr. 7.3 je zachycen postupný pokles počtu zpracovaných požadavků za jednotku sekundy. Na konci testu klesl počet pod hranici 20 požadavků za sekundu na obou serverech.

Výkonnostní test ve virtualizačním prostředí probíhal ve stejném duchu, jako předchozí, avšak vlivem špatné izolovanosti není chování klientů z průběhů grafů čitelné. Proto zde budou uvedeny pouze grafy s vybraným úsekem testu, viz obr. 7.4. Jelikož zátěž byla dimenzována tak, aby se odezva serveru pohybovala okolo hranice tří sekund, přičemž virtuální server neudržoval stabilní výkon, docházelo k častým výpadkům klientů. Počet zpracovaných požadavků v závěru testu klesl pod hodnotu 36 požadavků za sekundu. Po skončení testu bylo v databázi uloženo 522 944 e-mailových signatur.

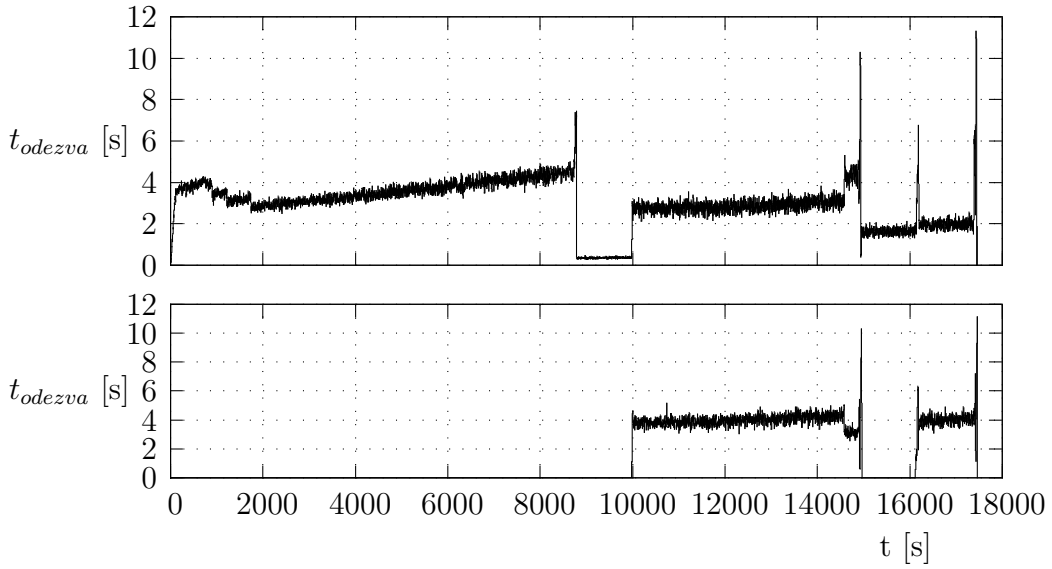


(a) průměrná doba odezvy

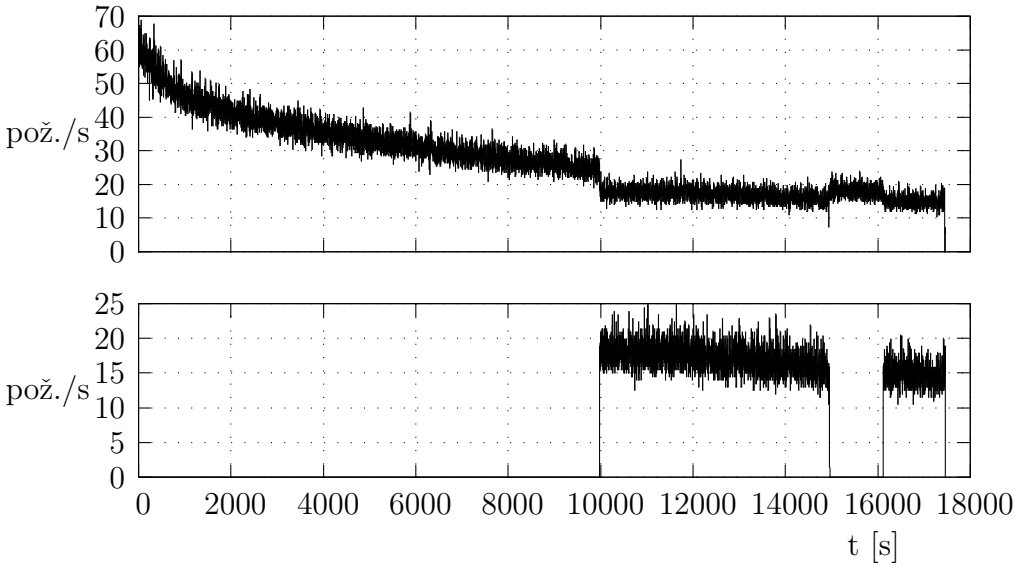


(b) propustnost

Obrázek 7.2: Grafické znázornění výsledků stresového testování II

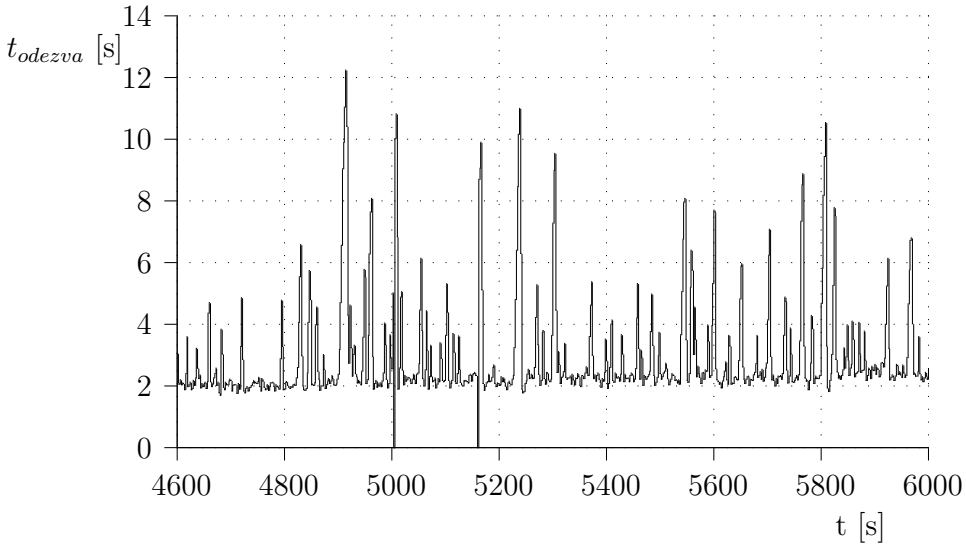


(a) průměrná doba odezvy

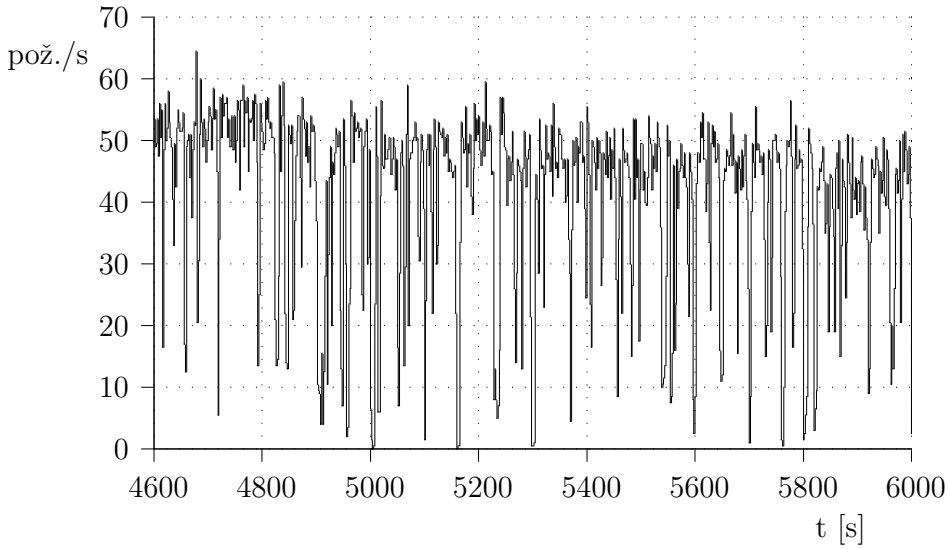


(b) propustnost

Obrázek 7.3: Grafické znázornění výsledků výkonnostního testování I



(a) průměrná doba odezvy



(b) propustnost

Obrázek 7.4: Grafické znázornění výsledků výkonnostního testování II

7.2.7 Kvalita signatury

Pro ověření chování signatury byl spuštěn test na vybrané části vzorku e-mailových zpráv [18] čítající 93 330 zpráv. Za použití funkce *SequenceMatcher.ratio()* programovacího jazyka Python byly porovnány obsahy e-mailových zpráv, jež filtr prohlásil za podobné. Z celkového počtu 1234 podobných zpráv filtr špatně rozhodl v 67 případech. Špatně detekované (false-positive) zprávy tedy tvoří 5,43% z celkového počtu detekovaných zpráv. Do výpočtů nebylo zahrnuto 187 detekovaných zpráv, které byly obsahově identické. Skutečný počet podobných zpráv ve vybraném vzorku nebyl znám.

Naměřené hodnoty potvrzují skutečnost, že signatury tvořené navrženým způsobem jsou náchylné na zobrazení zcela odlišných zpráv na velmi podobné otisky.

8 Škálovatelnost a bezpečnost

8.1 Škálovatelnost

S ohledem na výsledky stresových testů lze soudit, že úzkým místem systému je způsob použití centrální databáze, který snižuje škálovatelnost celého systému.

Možným řešením by bylo neukládat do databáze data, která si může aplikace serveru držet a spravovat v paměti sama. Jedná se především o informace o klientech. Tím se ovšem zamezí možnosti přepnutí se na jiný server bez nutnosti navázání spojení na webový server a přihlášení do systému. V tom případě by se přesunulo řízení výběru serveru ze serveru na klienta. Klient by si tak musel držet informace o serverech, se kterými již v minulosti komunikoval, a výkonnostních parametrech, díky kterým se mohl rozhodovat, jaký server zvolí pro následující komunikaci. Výhodou tohoto způsobu je volba geograficky či z hlediska konektivity lépe umístěných serverů vůči klientu.

Některá data si servery navzájem musí sdílet, či vyměňovat. Jde o blokované IP adresy a e-mailové signatury. Řešení problematiky tedy směřuje k použití distribuované databáze.

8.2 Bezpečnost

Komunikační vrstva, pro výměnu dat mezi klientem a serverem, je navržena tak, aby ze své podstaty byla komunikace bezpečná. Pakety obsahují sekvenční čísla, kontrolní součty a používá se šifrování. Úvahy o bezpečnosti komunikace byly uvedeny v kap. 5.3.4 a 5.3.6.

S ohledem na možné útoky typu Denial of Services (DoS) či Distributed Denial of Services (DDoS) je systém snadno napadnutelný. Při příjmu nadměrného množství požadavků se požadavky budou při překročení maximálních kapacit front zahazovat. Z toho vyplývá, že vyřazení serveru z provozu by trvalo stejně dlouho jako samotný útok. Úplný výpadek systému však nastat nemusí, a to v případě, že nebudou napadeny všechny servery systému. S postupem času totiž všichni klienti začnou své požadavky zasílat na servery, které zmíněným útokem napadeny nebyly.

Útok na enormní nárůst fronty je zabezpečen zmiňovaným definováním maximální délky fronty přijatých požadavků na serveru, resp. odpovědí na klientu. Při překročení jsou data zahazována.

Útok na nestandardní délku paketu je eliminován maximální délkou přijímaného paketu ze soketu. Následně se provádí kontrola délky přijatého paketu, aby se do front neukládaly přijatá data zbytečně. Na klientu se délka přijatého paketu musí přesně shodovat s délkou odpovědi. Na serveru musí být délka přijatých dat v rozmezí délek nejkratší a nejdelší verze paketu požadavku.

Pro zabezpečení proti podvrhu serveru se přihlášení klienta provádí užitím protokolu HTTPS. Klient ověřuje identitu serveru kontrolou platnosti certifikátu a shodou doménového jména. Pokud server neprojde zmiňovanou kontrolou, nebudou s ním vyměněny žádné informace.

Aby se na aktualizaci spamového skóre (četnosti výskytu) e-mailových zpráv podíleli jen důvěryhodní klienti, pracuje se s příznakem důvěryhodnosti uloženým u každého licenčního čísla v databázi.

Z hlediska bezpečnosti a ochrany dat týkajících se licenčních čísel je v případě zadání nekorektního licenčního čísla zablokována IP adresa, ze které se klient přihlašuje, na dobu pěti minut. Během této doby nesmí klient opakovat pokus o přihlášení. Učiní-li tak, je doba přenastavena na nových 5 minut.

Možným bezpečnostním nedostatkem je vysoké riziko špatné klasifikace zpráv, kde legitimní pošta je označena jako spam. Výsledky měření na testovacím vzorku zpráv jsou uvedeny v kap. 7.2.7. Použitý princip vytváření signatury neumožňuje výrazně snížit poměr false-positive zpráv. V případě potřeby by tak musela být do filtru nasazena jiná metoda stanovení charakteristické signatury zprávy.

9 Možnosti vylepšení

Programy v současné době pracují pouze s adresami IPv4. Rozšíření pro použití adres typu IPv6 by nevyžadovalo přílišný zásah do obou programů, klienta a serveru. Webový server by při přihlášení klienta musel zjistit typ adresování, které hodlá použít, a podle toho mu nabídnout odpovídající seznam serverů.

Struktura paketu odpovědi obsahuje vyhrazené místo pro řídící data, kterými server může ovlivňovat chování klienta. Systém zatím toto pole nevyužívá. Představou je použití řídících znaků pro snížení paralelizace dotazů generovaných klienty, v případě nadměrné zátěže serveru, eventuálně by server mohl generovat příznak vynucující přepojení klienta.

Systém si drží v databázi seznam serverů i s jejich aktuálním vytížením. Záznamy o serverech se automaticky přidávají či odebírají z databáze, když se aplikace serveru spouští a ukončuje. Pokud by server z nějakých nespecifikovaných důvodů násilně ukončil svoji činnost, záznam v databázi zůstane a server bude dále nabízen klientům. Zavedením časových značek poslední aktualizace stavu serveru by se mohla realizovat kontrola, kde by se staré záznamy jednoduše odstranily. Při odstranění záznamu z databáze se zároveň může generovat upozornění pro správce systému.

Další zlepšení se týká způsobu výpočtu spamového skóre. Současný systém pouze počítá četnost podobných signatur e-mailových zpráv. Spamové skóre by mohlo být upravováno vzhledem k úrovni důvěryhodnosti klienta a dosavadního spamového ohodnocení zprávy, které je vytvořeno ostatními filtry na serveru Kerio Connect. V současnosti je důvěryhodnost binárního typu, kdy je klient považován za důvěryhodného či nedůvěryhodného. Pro takto komplexněji zaměřený výpočet spamového skóre není v současnosti licenční politika společnosti Kerio Technologies s.r.o. vhodná.

Aplikace klienta přijímá odpovědi ze serveru na předdefinovaném případně při inicializaci přenastaveném konkrétním portu. Možným vylepšením by bylo zavedení povoleného rozsahu čísel portů, na které klient může naslouchat, případně delegovat určení portu operačnímu systému.

Posledním zmíněným, avšak neméně důležitým vylepšením, je portování na další platformy Mac a Windows. Při vývoji byly použity programové prostředky, které ve své podobě neumožňují portování. Jedná se především o použití knihovny pthread. Pro portování knihovny pthread na operační systém Windows lze použít knihovnu *pthread-win32*, avšak je nutné vytvořit strukturu *struct timespec*, která je použita pro časované čekání ve vláknech a není přítomna na operačním systému Windows.

10 Závěr

Cílem diplomové práce bylo navrhnout, implementovat a ověřit systém pro klasifikaci e-mailových zpráv podle četnosti výskytu a jejich typu. Výsledky provedených testů ověřily, že navržený systém se chová podle předpokladů zadání.

Princip anti-spamového filtru je založen na sběru a porovnávání signatur jednotlivých e-mailových zpráv. Filtr je realizovaný jako aplikace typu klient-server s distribuovaným serverem. Klient ze zadaných e-mailových zpráv stanovuje charakteristické signatury, které zasílá na server. Server klientům vrací četnost výskytu podobných signatur. Pro komunikaci mezi klientem a serverem byl navržen protokol, který umožňuje přihlášení klienta do systému a následně zabezpečuje přenos dat mezi klientem a serverem. Pro přihlášení je použit protokol HTTPS, kterým se mimo jiné vymění symetrický klíč pro šifrování dat ve zbytku komunikace, jež probíhá protokolem UDP. Aplikace obsahuje mechanizmy, které ošetřují možné chyby při doručení dat. Vytvořeným protokolem je zaručena bezpečnost komunikace, kdy se serverem mohou komunikovat pouze ověření klienti, a tím je předcházeno kompromitaci databáze signatur.

K tvorbě signatur e-mailových zpráv byl použit Goertzelův algoritmus, který pochází z oboru zpracování digitálního signálu a zaměřuje se na frekvenční analýzu digitálního signálu, jinak řečeno e-mailové zprávy. Díky předzpracování e-mailové zprávy lze získat následným užitím Goertzelova algoritmu amplitudy významných frekvencí, které tak tvoří jednotlivé části charakteristické signatury. Získané signatury jsou odolné proti záměnám znaků s menším skokem ASCII hodnoty, avšak větší změny ve zprávách nepostihnou.

Při testování funkčnosti se pracovalo s různými úrovněmi ztrátovosti paketů a byly vytvářeny nahodilé události jako výpadky serverů, linky či webového serveru. Zátěžovým testováním bylo zjištěno, že je filtr použitelný v prostředí s menším počtem klientů, neboť výsledky provedených výkonnostních testů odhalily úzké místo ve způsobu použití centrální databáze, který snižuje celkový výkon systému. Propustnost systému klesla pod hranici 40 vyřízených požadavků za sekundu, při přibližném počtu osmi set tisíc záznamů signatur uložených v databázi. Situace se dá zlepšit změnou způsobu použití databáze signatur, o čemž pojednává kap. 8.1.

Pro zvýšení efektivity a spolehlivosti filtru by mohl být implementován jiný způsob vytváření signatur, neboť tvorba signatury založená na frekvenční analýze zprávy nedosahuje takových výsledků jako řešení používané v jiných obdobných produktech. Možné vylepšení a navázání na diplomovou práci je spatřeno v zavedení výpočtu spamového skóre na serverech anti-spamového filtru, kde v souvislosti s plánovanou změnou licenční politiky firmy Kerio Technologies s.r.o. bude umožněno rozlišovat důvěryhodnost klientů na více stupních a tím vymezit úroveň, s jakou se klienti budou podílet na stanovení spamového skóre zprávy.

Seznam zkratek

AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
CBC	Cipher-block chaining
CFB	Cipher feedback
CPU	Central processing unit
DCC	Distributed Checksum Clearinghouses
DDoS	Distributed Denial of Service
DFT	Discrete Fourier transform
DNS	Domain Name System
DNSBL	Domain Name System Blocklist
DoS	Denial of Services
DSP	Digital signal processing
DTMF	Dual-Tone Multi-Frequency Signaling
EU	European Union
FFT	Fast Fourier Transform
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IMAP	Internet Message Access Protocol
IRC	Internet Relay Chat
ISP	Internet service provider
MD5	Message-Digest Algorithm
MDA	Mail Delivery Agent
MTA	Mail Transfer Agent
MUA	Mail User Agent
NAT	Network Address Translation
P2P	Peer-to-Peer
POP	Post Office Protocol
RAM	Random Access Memory
SHA1	Secure Hash Algorithm 1
SMTP	Simple Mail Transfer Protocol
SPF	Sender Policy Framework
SSH	Secure Shell
TCP	Transmission Control Protocol
UBE	Unsolicited Bulk Email
UCE	Unsolicited Commercial E-mail
UDP	User Datagram Protocol

USA ..... United States of America
USB ..... Universal Serial Bus

Literatura

[1] *OpenSSL: Documents, crypto(3)* [online]. [cit. 24.4.2012]. Dostupné z: `http://www.openssl.org/docs/crypto/crypto.html`.

[2] *Distributed Checksum Clearinghouses* [online]. [cit. 3.4.2012]. Dostupné z: `http://www.rhyolite.com/dcc/`.

[3] *libcurl - the multiprotocol file transfer library* [online]. Haxx. [cit. 23.4.2012]. Dostupné z: `http://curl.haxx.se/libcurl/`.

[4] *The XML C parser and toolkit of Gnome* [online]. [cit. 24.4.2012]. Dostupné z: `http://xmlsoft.org/`.

[5] *Connector C++ - MySQL Forge Wiki* [online]. [cit. 24.4.2012]. Dostupné z: `http://forge.mysql.com/wiki/Connector_C%2B%2B`.

[6] *Nilsimsa* [online]. [cit. 3.5.2012]. Dostupné z: `http://ixazon.dynip.com/~cmeclax/nilsimsa.html`.

[7] *About Pyzor* [online]. [cit. 2.5.2012]. Dostupné z: `http://sourceforge.net/apps/trac/pyzor/wiki/About`.

[8] *Vipul's Razor* [online]. [cit. 3.4.2012]. Dostupné z: `http://razor.sourceforge.net/`.

[9] *Vipul's Razor v2 README* [online]. [cit. 3.5.2012]. Dostupné z: `http://razor.sourceforge.net/docs/doc.php?type=text&name=README`.

[10] *Spam se mění a filtrace přestává být efektivní* [online]. [cit. 28.4.2012]. Dostupné z: `http://www.root.cz/clanky/spam-se-meni-a-filtrace-prestava-byt-efektivni/`.

[11] *The Definition of Spam* [online]. Spamhaus. [cit. 22.1.2012]. Dostupné z: `http://www.spamhaus.org/definition.html`.

[12] *CAN-SPAM Act of 2003* [online]. Wikipedia – the free encyklopedia. [cit. 25.2.2012]. Dostupné z: `http://en.wikipedia.org/wiki/CAN-SPAM_Act_of_2003`.

[13] *Spam* [online]. Wikipedia – the free encyklopedia. [cit. 12.2.2012]. Dostupné z: `http://cs.wikipedia.org/wiki/Spam`.

[14] *Spambot* [online]. Wikipedia – the free encyklopedia. [cit. 6.2.2012]. Dostupné z: `http://en.wikipedia.org/wiki/Spambot`.

[15] *Security Labs Report: January - June 2011 Recap.* White paper, M86 Security. Dostupné z: http://www.m86security.com/documents/pdfs/security_labs/m86_security_labs_report_1h2011.pdf.

[16] BANKS, K. *The Goertzel Algorithm* [online]. EE Times. [cit. 11.2.2012]. Dostupné z: http://www.eetimes.com/design/signal-processing-dsp/4024443/The-Goertzel-Algorithm.

[17] BRECHLEROVá, D. Direktiva 2002/58/EC (Direktiva o soukromí a elektronické komunikaci). In *Proceedings of the 11th International Conference on Systems Integration*, Prague, Czech Republic, June 16 - 17 2003. ISBN 80.

[18] COHEN, W. W. *Enron Email Dataset* [online]. Carnegie Mellon University. [cit. 20.3.2012]. Dostupné z: http://www.cs.cmu.edu/~enron/.

[19] DAMIANI, E. et al. An Open Digest-based Technique for Spam Detection. San Francisco, CA, USA, September 15-17 2004. Proc. of the 2004 International Workshop on Security in Parallel and Distributed Systems,.

[20] JEZEK, K. – HYNEK, J. The Fight against Spam - A Machine Learning Approach. In *ELPUB'07*, s. 381–392, 2007.

[21] KOZIEROK, C. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference.* No Starch Press, 2005. ISBN 1-59327-047-X.

[22] SCHRYEN, G. *Anti-spam measures - analysis and design.* Berlin Heidelberg New York : Springer, 2007. ISBN 978-3-540-71748-5.

[23] SPITZNER, L. *Honeypots: Definitions and Value of Honeypots.* White paper, May 2003. Dostupné z: http://www.tracking-hackers.com/papers/honeypots.html.

[24] SRIVASTAVA, T. V. *Phishing and Pharming – The Deadly Duo.* White paper, SANS Institute. Dostupné z: http://www.sans.org/reading_room/whitepapers/privacy/phishing-pharming-evil-twins_1731.

[25] STIRLING, S. *Quality Techniques Newsletter - Load Testing Terminology* [online]. Software Research, Inc. [cit. 19.4.2012]. Dostupné z: http://www.soft.com/News/QTN-Online/qtnsep02.html.

[26] WONG, M. – SCHLITT, W. Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1. RFC 4408 (Experimental), April 2006. Dostupné z: http://www.ietf.org/rfc/rfc4408.txt.

[27] ZHU, Z. et al. Botnet Research Survey. In *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, s. 967 –972, 28 2008-aug. 1 2008. doi: 10.1109/COMPSAC.2008.205.

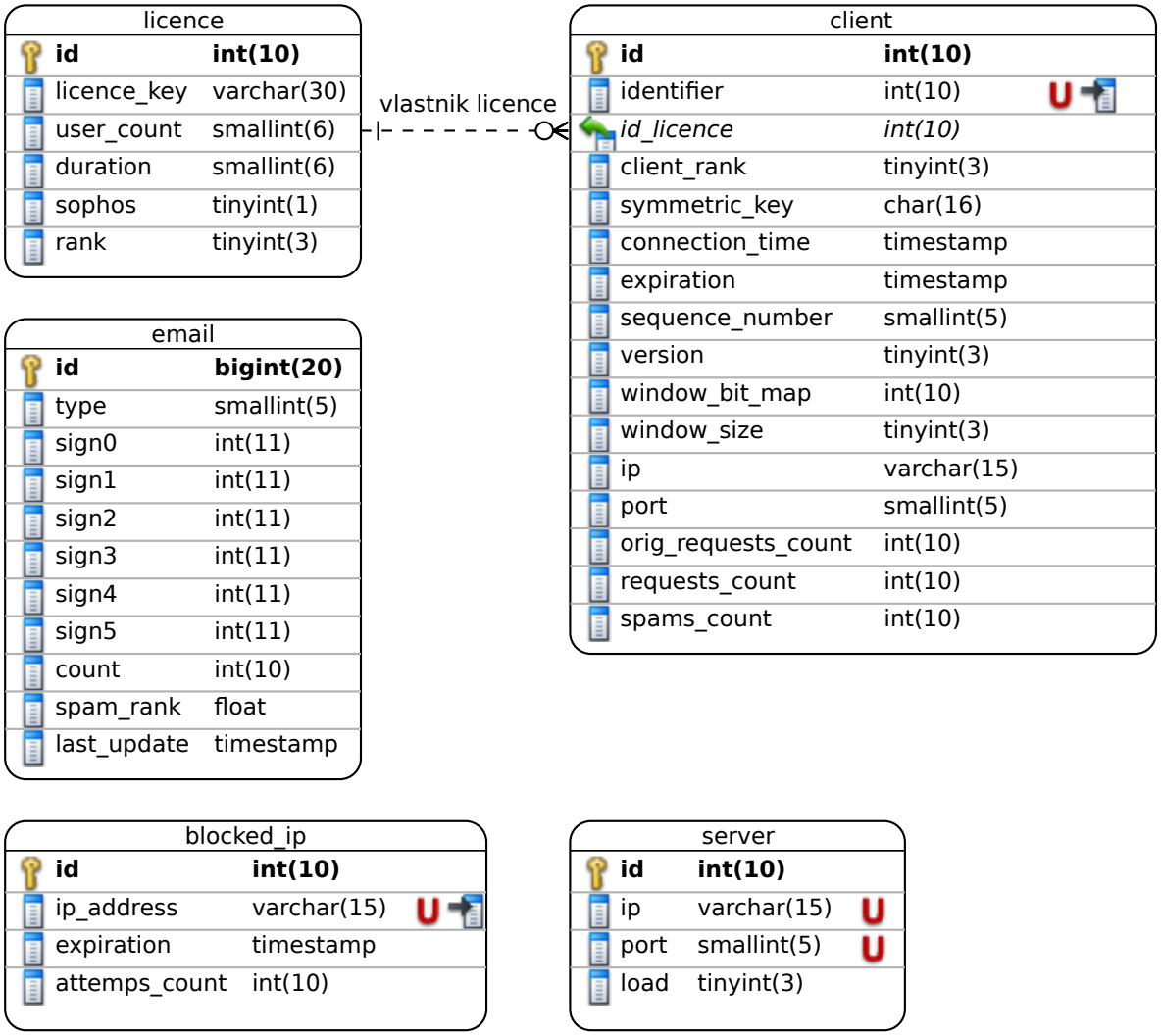
A Obsah přiloženého média

Složka	Obsah složky
doc/	text diplomové práce
dssd_client/	zdrojové soubory programu klienta
dssd_client/libdssd/	zdrojové soubory knihovny klienta
dssd_server/	zdrojové soubory programu serveru
email_corpus/	vzorek e-mailových zpráv
install/	instalační skripty
sql/	data databáze
test_results/	výsledky testů, převážně zátěžových
virtual_machine/	virtuální stroj s připraveným prostředím
web_frontend/	zdrojové kódy webového serveru

Tabulka A.1: Obsah složek na přiloženém médiu

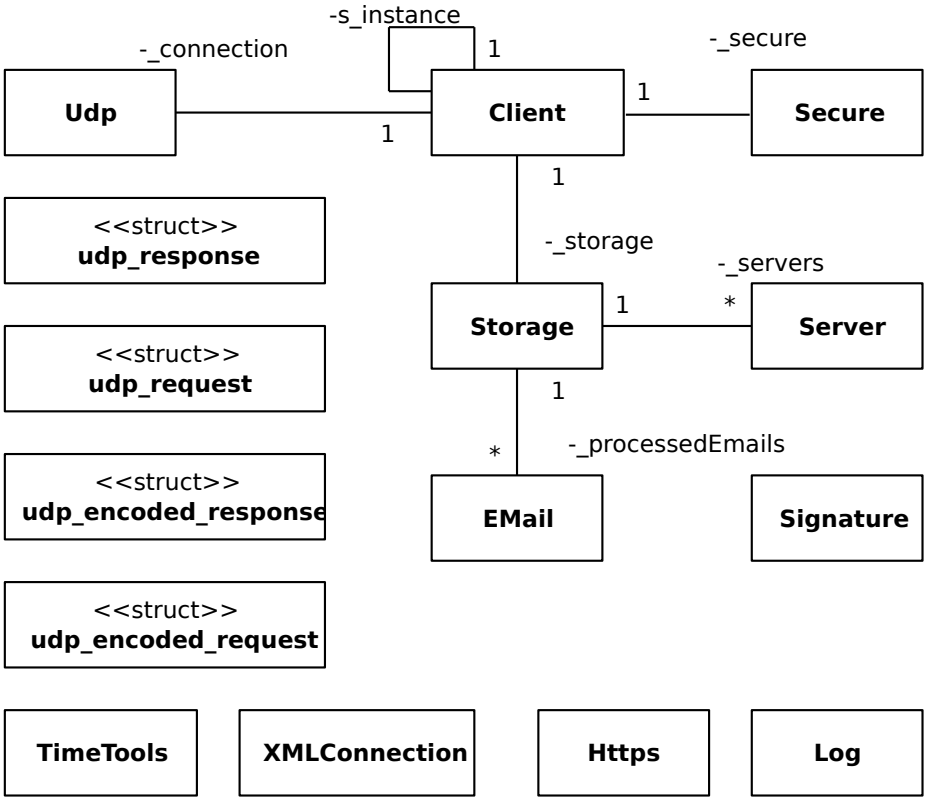
B Diagramy

B.1 ER diagram databáze



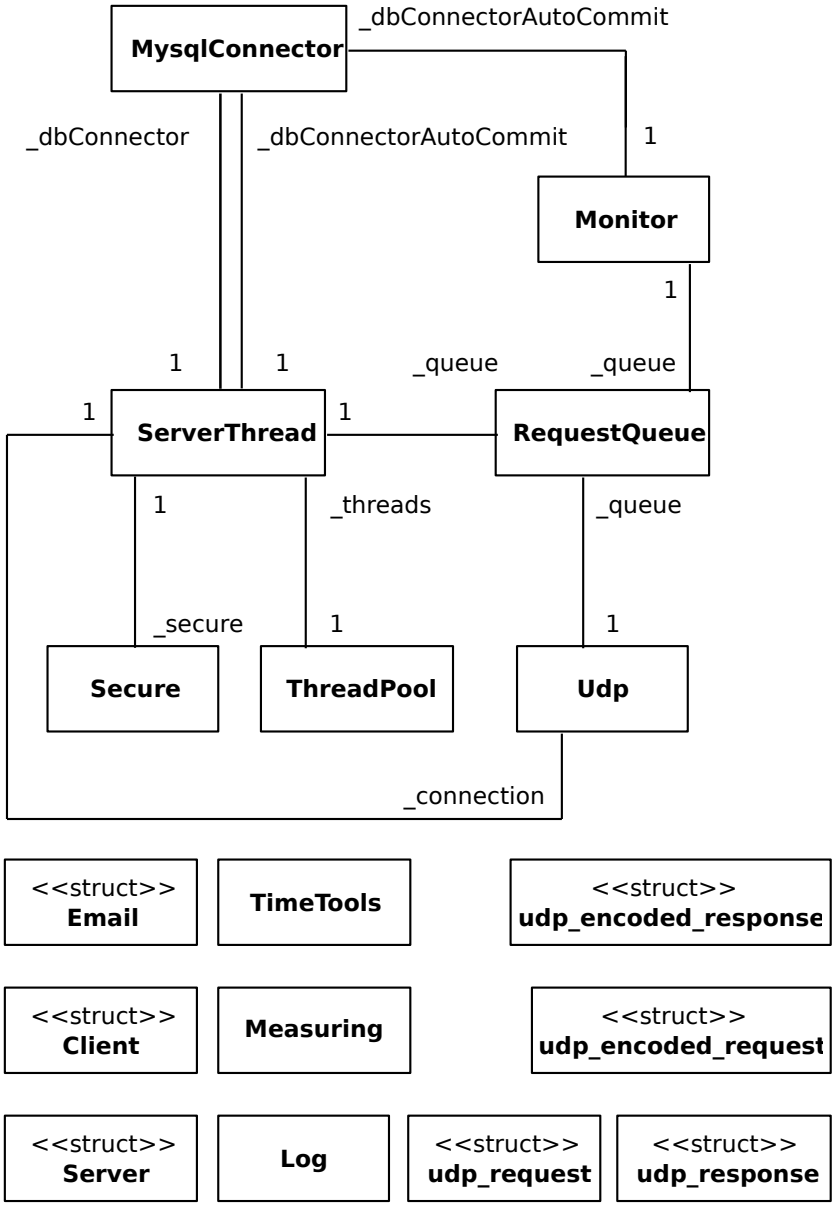
Obrázek B.1: ER diagram databáze

B.2 Diagram tříd knihovny klienta



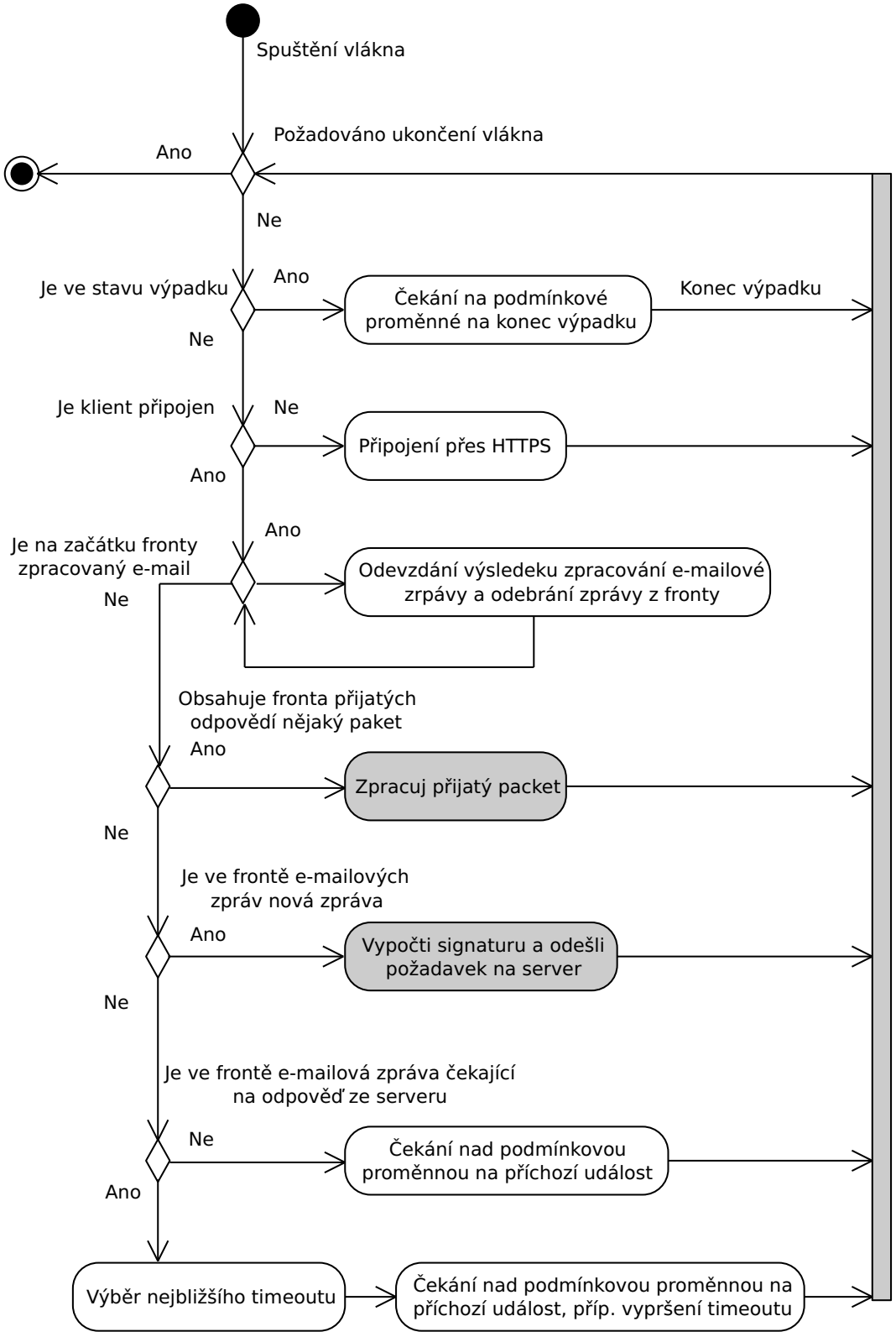
Obrázek B.2: Diagram tříd knihovny klienta

B.3 Diagram tříd programu serveru



Obrázek B.3: Diagram tříd programu serveru

B.4 Vývojový diagram vlákna klienta



Obrázek B.4: Vývojový diagram vlákna klienta

C Uživatelská příručka

Zprovoznit a ověřit funkčnost vytvořeného systému lze dvěma způsoby. První komplikovanější způsob v sobě zahrnuje kompletní přípravu a nastavení prostředí. Jedná se o instalaci programového vybavení, překlad programů, nastavení databáze a webového serveru.

Druhý způsob spočívá ve využití přiloženého virtuálního stroje, kde je již vše pečlivě připraveno. Použití virtuálního stroje je popsáno v kapitole C.2.

V obou případech je ovšem nutné zkopírovat obsah přiloženého média kamkoliv na pevný disk hostitelského systému či jiné úložiště s možností zápisu.

C.1 Příprava prostředí

Pro kompilaci, překlad a následné použití vytvořených programů, je potřeba nainstalovat následující programové vybavení a knihovny. Jelikož programy jsou platformě závislé, budou zde uvedeny návody pro platformu Unix, konkrétně pro operační systém GNU/Linux Debian 6.

C.1.1 Prostředí serveru

Před kompilací a sestavením programu serveru musí být nainstalováno následující programové vybavení a knihovny:

- g++
- libmysqlcppconn-dev
- libboost-dev
- libssl-dev

Jmenované programové vybavení lze nainstalovat pomocí balíčkového systému, přesněji řečeno pomocí programu *apt-get*. Pro automatickou instalaci potřebného programového vybavení je vytvořen skript *install/install-server-environment.sh*. Pro jeho spuštění je vyžadováno oprávnění uživatele root. Knihovna *libboost-dev* není v programu serveru použita, avšak její přítomnost je při překladu vyžadována knihovnou *libmysqlcppconn-dev*.

Bude-li zprovozněno více prostředí, ve kterých poběží programy serverů, je nutné tyto prostředí časově synchronizovat.

Databáze

Server vyžaduje ke své práci MySQL databázi. Ta se instaluje příkazem balíčkového systému *apt-get install mysql-server*. V případě nutnosti se před příkaz přidá *sudo*. Při instalaci databáze bude vyžadováno heslo uživatele root. Může být zadáno jakékoliv heslo. Natavení a naplnění databáze se provede vykonáním sql skriptu po přihlášení do mysql prostředí následujícím způsobem:

```
mysql -u root -p
source sql/database.sql
quit
```

Sql skript vytvoří nového uživatele *dssd* s heslem *asd#$%533* a s oprávněním přistupovat k databázi odkudkoliv. Vytvoří se databáze se všemi potřebnými tabulkami a uloží se do ní následující licenční klíče, kde první jmenovaný klíč nemá oprávnění k aktualizaci e-mailového skóre:

```
11111-11111-11111
11111-22222-33333
33333-22222-11111
```

V případě, že se na databázi bude připojovat další server z jiného stroje, je potřeba povolit přístup uživatele dssd do databáze z libovolné IP adresy.

Na závěr musí být databázový server restartován příkazem */etc/init.d/mysql restart*.

Webový server

Pro zprovoznění webového serveru je zapotřebí instalovat následující balíky:

- apache2
- php5
- libapache2-mod-php5
- php5-mysql
- php5-mcrypt

Je nutné nastavit apache tak, aby poskytoval bezpečné připojení. Není potřeba generovat nový certifikát a klíče. Lze použít stávající certifikát poskytovaný webovým serverem. Aplikace je nastavená tak, aby se spojila i se zabezpečenou stránkou s neověřeným certifikátem.

Zdrojové kódy webového stránky umístěné ve složce *web_frontend/* je potřeba zkopírovat do kořenové složky webových stránek na serveru */var/www/*, případně do jiné složky v závislosti na definovaných cestách v konfiguraci serveru. Jelikož webová aplikace vytváří log soubory do adresáře */var/www/log/* je nutné změnit vlastníka adresáře na uživatele *www-data*.

Záznamy v databázi se stávají s odstupem času zastaralé. Proto byl vytvořen PHP skript pro jejich automatické mazání, který je spouštěn periodicky podle cron tabulky. Do cron tabulky (*/etc/crontab*) je potřeba přidat následující záznam, který každý den v 05:00 hodin smaže zastaralé záznamy z databáze:

```
0 5 * * * www-data /usr/bin/php /var/www/script/cleanup.php
```

C.1.2 Prostředí klienta

Před kompilací a sestavením programu klienta se musí nainstalovat následující programové vybavení a knihovny:

- g++
- libcurl
- libssl-dev

- libxml++2.6-dev
- pkg-config

Mimo knihovny *libcurl*, která je umístěna ve složce *install/libs/curl7.24.0/*, se všechny jmenované programy a knihovny instalují pomocí balíčkového systému. Přiloženou knihovnu *libcurl* lze nainstalovat pomocí skriptu *install/libs/install-libcurl.sh*. Všechny prostředky potřebné klientem, lze instalovat skriptem *install/install-client-environment.sh*. Oba jmenované skripty potřebují oprávnění uživatele root.

Klient se snaží navázat HTTPS spojení na adrese *deb-dssd-server1.localdomain*. Proto je nezbytné do souboru */etc/hosts* přidat patřičný záznam, který bude definovat překlad doménového jména na IP adresu serveru. Pro použití virtuálního stroje má záznam následující podobu:

```
172.16.119.128  deb-dssd-server1.localdomain
```

C.2 Virtuální stroj

Pro usnadnění byl vytvořen virtuální stroj v programu VMware Player. Obraz virtuálního stroje je umístěn na přiloženém mediu ve složce *virtual_machine/*. Před jeho spuštěním zkopírujte obraz na pevný disk počítače.

Na virtuálním stroji je instalováno jak prostředí pro překlad klienta a serveru, tak i webový server s databází. Není na něm instalováno žádné grafické uživatelské rozhraní. Ke stroji je možno se připojit přes Secure Shell (SSH). Podrobné informace o virtuálním stroji jsou uvedeny v příloze E.

C.3 Překlad programů

Předpokladem pro úspěšné přeložení programů je řádně připravené prostředí, viz kap. C.1.1 a C.1.2.

Zdrojové kódy programů jsou umístěny ve složkách *dssd_client/* a *dssd_server/*, které obsahují i své soubory Makefile, definující překlad aplikací. Aplikace se zkompilují a sestaví použitím programu *make* v kontextu jednotlivých složek.

Server lze přeložit do módu s rozšířenou funkcí monitoringu, kde se měří výkonností parametry a jejich hodnoty jsou vypisovány na standardní výstup. Server se do zmíněného módu přeloží přidáním parametru při překladu *make EXTMONITORING=1*. Uvedená hodnota udává periodu měření výkonnostních parametrů, tzn. pro uvedený příklad se budou výkonnostní parametry vypisovat každou sekundu.

Překlad klienta se skládá ze dvou kroků, kde se v prvním kroku vytváří knihovna obsahující veškerou logiku klienta a následně v druhém kroku je tato knihovna použita pro sestavení spustitelného programu. Knihovna se generuje do složky *dssd_client/libdssd/dist/* a spustitelný program se ukládá do složky *dssd_clint/dist/*.

Klient lze přeložit do testovacího módu, ve kterém se nepočítají signatury e-mailových zpráv, nýbrž jsou generovány. Příkazem *make GEN=0* v kontextu složky klienta se přeloží klient do testovacího módu, kde zadané číslo udává počet generovaných zpráv. Nula značí neomezené množství generovaných zpráv.

Program serveru se po úspěšném překladu umist'uje do složky *dssd_server/dist/*.

C.4 Spuštění

Pro korektní běh programů je vyžadováno nastavené prostředí podle sekce C.1 v případě, že nebude použit virtuální stroj viz sekce C.2, kde je již požadované prostředí kompletně nastaveno.

Programy lze spustit z příkazové řádky příkazem *dssd_client*, případně *dssd_server*, v kontextu příslušné složky *dssd_client/dist/*, případně *dssd_server/dist/*. Chování programů se ovlivňuje pomocí parametrů zadaných při spouštění programu. Nápověda s detailním popisem parametrů je součástí přílohy viz sekce D.1 a D.2. Shodná nápověda lze vypsat uvedením přepínače *-h* při spuštění programu.

Program serveru při spuštění nevyžaduje zadání žádného parametru. Všechny parametry jsou volitelné. Naproti tomu klient vyžaduje zadání cesty k adresáři, ve kterém jsou umístěny e-mailové zprávy, jež bude zkoumat, nebo jedné či více cest vedoucích přímo ke zkoumaným e-mailovým zprávám. V případě zadání cesty k adresáři je vyžadováno uvést přepínač *-d* před samotnou cestou. Příklad jednoduchého spuštění aplikací:

```
./dssd_server
./dssd_client -d /home/jurecka/CD/email_corpus
```

V základu jsou nastaveny aplikace tak, aby nevytvářely žádné logovací záznamy. Přepínačem *-m* následovaným číselnou hodnotou logovací úrovně lze chování vytváření logů změnit. Jednotlivé úrovně logování lze kombinovat sečtením jejich číselných hodnot. Pokud nebude přepínačem *-o* specifikován výstupní soubor, budou logovací záznamy zapisovány na standardní výstup.

Výstupem programu *dssd_client* jsou trojice skládající se z názvu souboru zkoumané e-mailové zprávy, počtu zpráv vyhodnocených serverem jako podobných a spamového skóre. Vztah mezi četností a spamovým skórem je podrobněji popsán v kap. 5.4.1.

Programy je možno ukončit klávesovou zkratkou CRTL+C, kde po jejím stisku dojde k dokončení právě prováděné operace a ukončení aplikace. Aplikace serveru navíc odebere svůj záznam z databáze, a tak dále nebude tento server nabízen klientům.

Příklad spuštění aplikací se zadanými parametry:

```
./dssd_server -a 172.16.119.128 -p 30001 -m 1023 -o dssd_server.log \
 -d 192.168.160.128 -u dssd -s asd#$%533 -r 3306
./dssd_client -a 172.16.119.1 -p 20011 -l 11111-22222-33333 -m 1023 \
 -o dssd_client.log -d /media/data/vbox_share/enron_inbox/
```

V případě, že klient se serverem nejsou schopni komunikovat, je zapotřebí zkontrolovat log výstupy, ze kterých by mělo být patrné, co způsobuje daný problém. Především log výstupy týkající se komunikační vrstvy můžou naznačit skutečnost, že jsou na některém z firewallů nevhodně nastavená pravidla, která omezí přenos UDP paketů.

C.4.1 Dávkové spuštění klientů

Pro usnadnění testování systému byly vytvořeny skripty pro spuštění a ukončení N klientů na pozadí. Při spuštění klientů skriptem *runner.sh* jsou vyžadovány parametry specifikující počet spouštěných klientů, počet klientů spouštěných naráz v jedné dávce, časovou prodlevu mezi jednotlivými dávkami v sekundách a složku obsahující e-mailové zprávy. Skript *runner-stats.sh* je rozšířením skriptu *runner.sh*, kdy po spuštění posledního klienta se automaticky ukončí běh všech klientů skriptem *killer.sh*.

Příklad postupného spuštění celkového počtu sta klientů v dávkách po pěti klientech se vzájemným časovým odstupem dvě sekundy:

```
./runner.sh 100 5 2 /home/jurecka/CD/email_corpus
```

Ukončení všech běžících klientů se docílí voláním následujícího skriptu:

```
./killer.sh
```

D Nápověda programů

D.1 Klient

```
NAME
    dssd_client - determine spam score of given email messages

SYNOPSIS
    dssd_client  [-hs] [-a bind_address] [-l licence_key ] [-m log_mask ]
                 [-o output_log_file ] [-p bind_port ]
                 {-d source_directory | file ...}

DESCRIPTION
    dssd_client is a program for determination spam score of email messages.
    Program calculate email signatures and send them to a remote server.
    Spam scores are assigned from server's responses. Program print on
    standard output path to the email file, count of similar emails and
    calculated spam score.

    Researched email messages are designated by file severally
    or by source_directory collectively.

    The options are as follows:

    -a bind_address
            Specifies an address of local machine as source address
            of an connection. Useful in case of multi-address machine.

    -d source_directory
            Sets directory from which is taken email messages.

    -h      Print help.

    -l licence_key
            Specifies licence key for authentication to server.

    -m log_mask
            Sets the log mask defining log particularity. It can be set any
            combination of following log types. Set the summary value
            of the chosen types.
            LogOff      0    no log records
            LogAuth     1    authentication process records
            LogClient   2    client thread records
            LogSecure   4    security records
            LogNetwork  8    network traffic records
            LogSigns    16   signature determination records
            LogWarning  512  print all warning logs
            LogError    1024 print all error logs
            LogAll      2047 print all log records

    -o output_log_file
            Specifies a log file for append log records. If not set, the log
            records are printed to the standard output.

    -p bind_port
            Specifies a port for listening server's responses. In default is
            used port 30000.

    -r      Recursive throughway of the source_directory.

    -s      Disable parallel requesting.
```

D.2 Server

```
NAME
    dssd_server - server of system to determination spam

SYNOPSIS
    dssd_server  [-h] [-a bind_address ] [-d database_address ] [-m log_mask ]
                 [-o output_log_file ] [-p bind_port ] [-r database_port ]
                 [-s database_password ] [-u database_user ]

DESCRIPTION
    dssd_server is a server part of the program for determination spam score
    of email messages. Program receive email signatures requests from client
    and send responses with spam score them back.

    The options are as follows:

    -a bind_address
            Specifies an ip address of local machine as source address
            of an connection. Useful in case of multi-address machine.

    -d database_address
            Specifies an ip address of a database server.

    -h      Print help.

    -m log_mask
            Sets the log mask defining log particularity. It can be set any
            combination of following log types. Set the summary value
            of the chosen types.
            LogOff      0    no log records
            LogAuth     1    authentication process records
            LogServer   2    Server threads records
            LogSecure   4    security records
            LogNetwork  8    network traffic records
            LogMonitor  16   monitor records
            LogQueue    32   queue of received request records
            LogDB       64   database records
            LogMeasuring 128 print measured values
            LogStats    256  interesting statistical values
            LogWarning  512  print all warning logs
            LogError    1024 print all error logs
            LogAll      2047 print all log records

    -o output_log_file
            Specifies a log file for append log records. If not set, the log
            records are printed to the standard output.

    -p bind_port
            Specifies a port for listening server's responses. In default is
            used port 30000.

    -r database_port
            Specifies a port of a database server.

    -s database_password
            Specifies a database password.

    -u database_user
            Specifies a database user.
```

E Virtuální stroj

E.1 Informace o virtuálním stroji

Uživatelské účty	
root	a
jurecka	a
Uživatelské účty databáze MySQL	
root	a
dssd	asd#$%533
Umístění	
přiloženého média	/home/jurecka/CD/
webového serveru	/var/www/
Síťové rozhraní	
eth0 (NAT)	172.16.119.128
eth1 (Host-only)	192.168.160.128

Tabulka E.1: Infromace o virtuálním stroji

F Specifikace testovacích prostředí

F.1 Prostředí pro přímé testování

Notebook - client	
Procesor	Intel Core i3-330M 2.13 GHz (4 jádra)
Operační paměť	2x2GB DDR3 1066MHz
Pevný disk	640GB SATA (5400rpm)
Operační systém	Debian 6.3.0 64bit
PC - server	
Procesor	Intel Core i5-2500K 3.3 GHz (4 jádra)
Operační paměť	2x2GB DDR3 1600MHz
Pevný disk	60GB SSD SATA3.0
Operační systém	Debian 6.3.0 64bit

Tabulka F.1: Specifikace prostředí pro přímé testování

F.2 Virtualizované prostředí

Virtuální stroj - klient	
Procesor	Intel Core2 6600 2.40 GHz (2 jádra)
Operační paměť	1GB
Operační systém	Debian 6.3.0 64bit
Virtuální stroj - server	
Procesor	Intel Xeon 3.73 GHz (2 jádra)
Operační paměť	2GB
Operační systém	Debian 6.3.0 64bit

Tabulka F.2: Specifikace virtuálního prostředí