VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

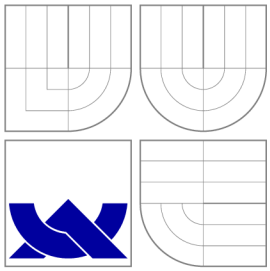
EMAILOVÝ SERVER JAKO SLUŽBA PRO SYSTÉMY ZALOŽENÉ NA TECHNOLOGII WINDOWS-NT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

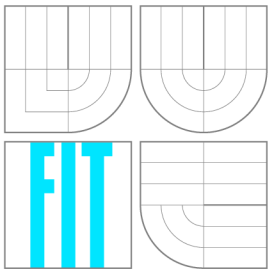
AUTOR PRÁCE
AUTHOR

Bc. PETR JALŮVKA

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

EMAILOVÝ SERVER JAKO SLUŽBA PRO SYSTÉMY ZALOŽENÉ NA TECHNOLOGII WINDOWS-NT

EMAIL SERVER AS A WINDOWS NETWORK SERVICE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PETR JALŮVKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL OČENÁŠEK
SUPERVISOR

BRNO 2007

Zadání diplomové práce

Řešitel	**Jalůvka Petr, Bc.**
Obor	Informační systémy
Téma	**Emailový server jako služba pro systémy založené na technologii Windows-NT**
Kategorie	Počítačové sítě

Pokyny:

1. Seznamte se s protokoly SMTP, POP3 a IMAP, případně s jejich zabezpečenými variantami(POP3s, IMAPs, ...), podle příslušných RFC
2. Seznamte se s možnostmi tvorby aplikací v prostředí Windows-NT(resp. Win-XP). Především se zaměřte na implementaci služeb (services)
3. Navrhněte vlastní emailový server umožňující práci s uvedenými protokoly. Server bude umět pracovat s lokálními emailovými složkami (víceuživatelský přístup) a případně předávat požadavky na vzdálené emailove servery.
4. Navržený server implementujte jako službu pro prostředí Win-NT.
5. Zhodnoťte dosažené výsledky a diskutujte další možnosti rozšíření.

Literatura:

- Příslušná RFC pro jednotlivé protokoly a formáty zpráv
- www.imap.org

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 – 3

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese
`http://www.fit.vutbr.cz/info/szz`

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí	**Očenášek Pavel, Ing.**, UIFS FIT VUT
Datum zadání	28. února 2006
Datum odevzdání	22. května 2007

Licenční smlouva

Licenční smlouva v kompletním znění je uloľena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Výňatek z licenční smlouvy:

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýděleč̌ně uľít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoľenin.

2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.

3. Autor souhlasí se zvěřejněním díla v databázi přístupné v mezinárodní síti:

   ☐ ihned po uzavření této smlouvy

   ☐ 1 rok po uzavření této smlouvy

   ☐ 3 roky po uzavření této smlouvy

   ☐ 5 let po uzavření této smlouvy

   ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsaľených informací).

4. Nevýděleč̌né zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyľaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Abstrakt

Email je jednou z nejpoužívanějších služeb na dnešním Internetu. Téměř neexistuje uživatel, který by neměl alespoň jednu vlastní schránku. Tyto schránky jsou ale většinou umístěny na cizích serverech. Pokud se uživatel rozhodne si vyzkoušet provozovat vlastní emailový server, musí si většinou nainstalovat unixový systém a následně podstoupit složitou konfiguraci emailového serveru. Cílem této práce je seznámit čtenáře s historií a základními protokoly pro emailovou komunikaci a dále navrhnout a implementovat emailový server pro platformu Windows NT, jenž bude lehce konfigurovatelný.

Klíčová slova

e-mail, mail, server, POP3, IMAP4v1, MIME, SMTP, .Net Framework, Windows-NT

Abstract

Email is one of the most used services on the Internet. It's hard to find a user with less than one own mailbox. These mailboxes are usually located on servers controlled by someone else than mailbox owners. Users trying to create their own email serverhave to install unix system and go through hard configuration process of the email server. Purpose of this work is to familiriaze reader with email history and protocols and then introduce design and implementation of an email server with easy configuration for Windows NT platform.

Keywords

e-mail, mail, server, POP3, IMAP4v1, MIME, SMTP, .Net Framework, Windows-NT

Citace

Petr Jalůvka: Emailový server jako služba pro systémy založené na technologii Windows-NT, diplomová práce, Brno, FIT VUT v Brně, 2007

Emailový server jako služba pro systémy založené na technologii Windows-NT

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Pavla Očenáška. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Jalůvka
21. května 2007

© Petr Jalůvka, 2007.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

Při tvorbě této kapitoly bylo využito [15].

Email je mnohem starší než Internet a dokonce i ARPANet, předchůdce Internetu. Nebyl vynalezen, vyvinul se z velmi jednoduchých počátků.

Raný email bylo pouze malé vylepšení toho, co dnes známe pod pojmem adresář - pouze umisťoval zprávu do adresáře jiného uživatele, kde jej mohl vidět při přihlášení. Jednoduše, jako bychom zanechali na stole psanou poznámku.

Zřejmě prvním emailovým systémem tohoto typu byl MAILBOX, jež používali na MIT (Massachusetts Institute of Technology) od roku 1965. Dalším programem pro zasílání zpráv na stejném počítači se jmenoval SNDMSG.

Počty uživatelů na některých sálových počítačích z této doby mohly dosahovat až stovek - uživatelé často pro přístup na hlavní počítač ze své kanceláře používali tzv. „dumb terminály“, což byly terminály bez jakéhokoliv vlastního úložného prostoru a paměti; veškerá výpočetní práce se prováděla na vzdáleném sálovém počítači.

Předtím, než byly počítače spojovány do sítí, tak mohl email sloužit pouze k zasílání zpráv uživatelům na stejném počítači. Ve chvíli propojení počítačů do sítí se stal problém ještě komplikovanějším. Bylo potřeba umístit zprávu do obálky a následně ji doručit. K tomu, aby bylo toto doručení možné, však bylo potřeba dát elektronickému systému vědět, podobně jako při odesílání dopisu, kam se má daná zpráva doručit. Tady se prvně začíná objevovat pojem adresa.

Přesně toto je důvod, proč je za duchovního otce emailu považován Ray Tomlinson. Podobně jako mnoho ostatních tvůrců Internetu, Tomlison pracoval pro BBN (zkratka jmen zakladatelů Bolt, Beranek, Newman). Použil symbol @ k označení přeposlání zprávy z jednoho počítače na jiný. Pro každého, jež tehdy užíval standarty Internetu, bylo tedy velmi jednoduché vytvořit danému uživateli adresu ve tvaru uživatelské_jméno@jméno_počítače. Jedním z prvních uživatelů tohoto systému byl Jon Postel. Tato osoba je spojena s popisem této služby jako „pěkný hack“. Ve skutečnosti to tak opravdu i bylo, což je dokázáno tím, že tento formát adres se užívá dodnes.

I přesto, co vše mezinárodní síť nabízela, zůstal email nejdůležitější a nejrozšířenější službou, která na Internetu byla. Počet uživatelů se tou dobou rozrostl na více než 600 miliónů.

V roce 1974 měl email stovky vojenských uživatelů, protože jej podporoval i ARPANet. Email se stal zachráncem Arpanetu a znamenal radikální posun v pohledu na jeho smysl.

Od této doby se rychlost vývoje rapidně zvýšila. Pro svého nadřízeného vyvinul Larry Roberts emailové složky, jež umožňovaly třídění emailů. To znamenalo obrovskou výhodu při jeho používání. V roce 1975 vyvinul John Vital software pro organizaci emailů, v roce

1976 se začaly objevovat první komerční balíčky pro užití emailu. Během několik let se email stál dominující službou Arpanetu, kdy představoval 75% veškerého objemu dat.

Email nás takto dovádí od Arpanetu k Internetu. Zde byl něčím, co chtěli využívat obyčejní lidé na celém světe.

V několika následujících letech docházelo v oblasti emailových aplikaci k tak rychlému a neuspořádanému vývoji, že vyjmenování všech osob, které se do tohoto dění nějakým způsobem zapojily, by byl nadlidský úkol.

I přesto byl tento vývoj velmi užitečný a to především v zemích, kde připojení se na nejbližší emailový systém skrze telefonní linku bylo velmi drahé (často to totiž v počátcích vedlo k mezinárodním hovorům). S cenou připojení mnoha dolarů za minutu záleželo také na možnosti si zprávu připravit před připojením, což v počátcích nebylo zcela obvyklé. Tento tzv. „offline“ mód nabízel také mnohem lepší a přátelštější rozhraní. Připojení přímo k emailové službě v této době, kdy existovalo jen velmi malé množství standardu, totiž často vyústilo v nutnost dalších operací, z důvodu např. špatného vykreslení textu emailu mimo obrazovku uživatelova počítače apod. V tomto směru byl offline mód bez nutnosti připojení opravdovou spásou pro uživatele.

Prvním významným standardem, jež je v upravené podobě užíván i dnes, byl SMTP (zkratka od Simple Mail Transfer Protocol - jednoduchý protokol pro přenos pošty). Tento protokol byl tak jednoduchý, že se ani nesnažil nějakým způsobem zjistit, zda odesílatel emailu je opravdu osobou, za níž se v emailu vydává. Bylo (a stále je) tak velmi jednoduché, zaslat email cizím jménem. Tyto chyby v protokolu pak byly později zneužity mnohými viry a červy, podvodníky a osobami šířícími nevyžádanou poštu. Většina chyb byla postupem času ošetřena, ale stále se SMTP nedá považovat za bezchybný.

Po definování tohoto protokolu se začaly objevovat jeho první komerční systémy. Jedním z prvních opravdu dobrých byl Eudora z roku 1988, jejímž autorem byl Steve Dorner. Tento systém se objevil krátce po uvedení dalšího dobrého systému, Pegasus mail.

Dalším důležitým standardem pro email byl POP (Post Office Protocol). Ještě před jeho schválením, kdy nebyly specifikace protokolu úplné, se začaly objevovat komerční systémy, což znamenalo, že každý systém se mírně odlišoval. Až po schválení tedy byl umožněn návrh vzájemně spolupracujících systémů.

Toto bylo období kdy telekomunikační společnosti zpoplatňovaly přístup na Internet minutovými sazbami, což vysvětluje návrh POP protokolu jako protokolu pro čtení emailů v offline módu, kdy si uživatel stáhl své emaily, následně se odpojil a mohl si číst svou poštu bez ohledu na čas. V tomto období byly také velmi populární emailové diskuzní skupiny. Na Internetu existovalo obrovské množství těchto skupin na různorodá témata. Dnes jsou na Internetu známy jako Usenet.

Postupem času, kdy se stalo připojení k Internetu běžnou a levnou záležitostí, se objevil další standart, IMAP (zkratka Interim (později Internet) Message Access Protocol), jež umožňoval práci s emaily bez nutnosti si je stahovat do svého počítače.

Email se stal dostupný široké uživatelské veřejnosti, začaly se objevovat „freemailové“ servery, jež umožňovaly uživatelům zdarma si založit a spravovat svou vlastní emailovou schránku pomocí přátelského webového rozhraní. Nic už tedy nebránilo jeho masovému rozšíření.

Kapitola 2

MIME

Formát textových zpráv přenášených v Internetu byl definován dvojicí norem RFC821 a RFC822 a následně aktualizován v RFC 2821[12] a RFC2822[18].

Brzy se však ukázalo, že tento standard nevyhovuje požadavkům uživatelů, protože se mail ukázal jako vhodný prostředek pro posílání nejen holých textových zpráv, ale i pro posílání obrázků, zvuků, formátovaného textu, obecně binárních souborů. V poslední době pak přibyl požadavek na posílání bezpečných zpráv, tj. šifrovaných i elektronicky podepsaných.

Princip výměny zpráv se rychle rozšířil i za hranice elektronické pošty. Zprávy podobného formátu používá nejenom elektronická pošta, ale i news a protokol HTTP.

Proto bylo potřeba vytvořit normy rozšiřující původní standard, jež se označují MIME.

Rozšíření MIME (Multipurpose Internet Mail Extension) je standardizováno normami RFC2045 až RFC2049 [5]. Zavedení MIME se snaží řešit omezení původního standardu podle RFC(2)822. MIME je standardem, který doplňuje RFC(2)822 a zajišťuje zpětnou kompatibilitu. Je navrženo tak, aby mohly být posílány stávajícím poštovním systémem zprávy obsahující diakritiku, obrázky, zvuk apod.

Tento standard řeší dvě otázky:

- Jak vytvořit ze zprávy obsahujícího např. binární data zprávu vyhovující RFC822 a tedy přepravitelnou používanými přenosovými protokoly. Tj. zavádí standard pro kódování.
- Jak rozlišit jednotlivé druhy zpráv, tj. zavádí klasifikaci přenášených informací. Klasifikace přenášených informací se ukázala velmi užitečnou i mimo e-mail. Moderní služby Internetu ji přebírají a používají ke stejnému účelu. MIME rozšíření zavádí nové hlavičkové řádky do mailové zprávy, které specifikují typ posílaných dat a způsob jejich kódování.

MIME zavádí oproti RFC(2)822 nové hlavičky :

- MIME-Version – přítomnost této hlavičky v mailu indikuje, že je zpráva sestavena podle RFC2045 až RFC2049.
- Content-Type – specifikuje typ a podtyp dat posílaných v těle zprávy (text, audio, video, virtuální realita).
- Content-Transfer-Encoding – specifikuje použité kódování, pomocí kterého je zpráva převedena do formátu vyhovujícímu přenosovému mechanismu (do ASCII).

- Content-ID – identifikace zprávy použitelná v možném odkazu
- Content-Description – textový popis obsahu.
- Content-řetězec – je rezervováno pro budoucí použití v MIME.

Kapitola 3

POP3

3.1 Popis

POP3 (Post Office Protocol version 3) je internetový protokol, který se používá pro stahování emailových zpráv ze vzdáleného serveru na klienta. Jedná se o aplikační protokol pracující přes TCP/IP připojení. POP3 protokol byl standardizován v roce 1996 v [3].

POP3 je následníkem protokolů POP1 a POP2 (označení POP už dnes téměř výhradně znamená POP3). V současné době používají téměř všichni uživatelé elektronické pošty pro stahování emailů programy využívající POP3. POP3 je výhodný pro uživatele, kteří mají časově omezené připojení k Internetu (jako je např. telefonní připojení). Zprávy si ze svého poštovního serveru pouze stáhnou, odpojí se a můžou si číst doručené emaily libovolně dlouho a být při tom offline (odpojen od Internetu). Menší nevýhodou (ale pro někoho to může být i výhodné) je fakt, že po přenesení zpráv ze serveru do klienta (tzn. počítače, do kterého budete stahovat zprávy) se na serveru tyto zprávy smažou. Jistá výhoda může být v tom, že pokud dostáváte mnoho pošty nebo velké emaily a nemáte na serveru velký prostor pro zprávy, tak budete mít všechny zprávy pouze doma na počítači a na poštovním serveru bude vždy místo pro další. Nicméně většina poštovních klientů umožňuje zanechání zprávy na serveru.

Ne každému uživateli bude asi vyhovovat fakt, že si nemůže vybrat, jaké zprávy se mu stáhnou. Stáhnou se mu všechny zprávy, třeba i ty, které uživatel číst nechce, nebo SPAM (pokud ho již nefiltruje poštovní server).

Bližší informace k protokolu jsou k nalezení v [3], příp. [4].

3.2 Zabezpečení

Jako mnoho jiných starších internetových protokolů, POP3 původně podporoval jenom nešifrované přihlašovací mechanismy. Ačkoli v POP3 je běžný jednoduchý (nezabezpečený) přenos hesel, podporuje současně několik metod ověřování k provedení proměnlivých úrovní ochrany před nezákonnými vstupy do cizího emailu.

Jsou definovány příkazy pro autentizaci APOP, který užívá MD5 hash funkci pro zabezpečený přenos hesla od klienta na server, a AUTH, jež definována v [1]. Používá stejný princip, jako tato metoda v IMAP protokolu.

Dále existuje také možnost šifrovat celou komunikaci pomocí TLS [14].

3.3 Komunikace

Protokol POP3 má pro své účely vyhrazen TCP port 110. Komunikace probíhá na principu výměny zpráv mezi klientem a serverem. Příkaz vždy začíná na začátku řádky, v základní implementaci POP3 mají příkazy 3 nebo 4 znaky. Příkazy nerozlišují velká a malá písmena. Za příkazem můžou následovat další argumenty, oddělené mezerami. Řádky jsou oddělovány pomocí CRLF. Každá odpověď od serveru musí začínat indikací stavu operace - buď +OK, nebo -ERR. Následovat může textový řetězec s popsaným důvodem stavu. POP3 implementace jsou často poměrně komunikativní a dají se užívat i „ručně“, např. pomocí telnetu.

3.3.1 Stavy komunikace

- Autorizace – tento stav nastává po otevření spojení a kladné odpovědi od serveru. Server očekává autentizaci klienta a po jejím úspěšném provedení povolí klientovi přístup k požadované emailové schránce, již server uzamkne pro exkluzivní přístup. Pokud se nepodaří získat exkluzivní přístup, může server odeslat zápornou odpověď a spojení uzavřít.
- Transakce – v tomto stavu má klient výhradní přístup ke své schránce a může provádět veškeré operace s jejím obsahem. Každý příkaz od klienta je následnován odpovědí serveru.
- Aktualizace – pokud klient zadá příkaz QUIT ve stavu transakce, přejde server do stavu aktualizace obsahu, kdy se stanou veškeré změny, jež uživatel ve stavu transakce provedl, permanentními.

3.3.2 Odpovědi serveru

- +OK – kladná odpověď
- -ERR – záporná odpověď

3.3.3 Příklad komunikace

Příklad komunikace je uveden na obr. 3.1

3.4 Minimální požadavky na implementaci

Pro správnou funkčnost je dle [3], kde je k nalezení i popis funkčnosti, požadována podpora těchto příkazů :

- STAT, LIST, RETR, DELE, NOOP, RSET, QUIT

```
S: <server naslouchá na TCP portu 110>
C: <otevření spojení>
S:    +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C:    APOP mrose c4c9334bac560ecc979e58001b3e22fb
C:    USER mrose
S     +OK User accepted
C:    PASS mrosepass
S     +OK Pass accepted
S:    +OK mrose's maildrop has 2 messages (320 octets)
C:    STAT
S:    +OK 2 320
C:    LIST
S:    +OK 2 messages (320 octets)
S:    1 120
S:    2 200
S:    .
C:    RETR 1
S:    +OK 120 octets
S:    <POP3 server posílá 1. zprávu>
S:    .
C:    DELE 1
S:    +OK message 1 deleted
C:    RETR 2
S:    +OK 200 octets
S:    <POP3 server posílá 2. zprávu>
S:    .
C:    DELE 2
S:    +OK message 2 deleted
C:    QUIT
S:    +OK dewey POP3 server signing off (maildrop empty)
C:  <uzavření spojení>
S:  <server čeká na další spojení>
```

Obrázek 3.1: Příklad komunikace

Kapitola 4

IMAP

4.1 Historie

IMAP (Interim Message Access Protocol) byl vytvořen jako protokol pro prácí s emailovými schránkami na vzdáleném serveru.

Původní IMAP byl implementován jako klient Xerox Lisp Machine a TOPS-20 server. Žádná z těchto implementací se nedochovala, všechny byly aktualizovány na IMAP2. I přesto, že některé z příkazů a odpovědí byly podobné verzi IMAP2, nebyly kompatibilní.

IMAP2 byl první veřejnou verzí protokolu IMAP. Jeho popis byl publikován v [6], následně byl aktualizován v [7].

Další verzí byl IMAP2bis. Nová verze přinesla oproti verzi IMAP2 podporu pro MIME a funkce pro správu emailové schránky, jako jsou např. vytvoření, mazání či přejmenování. Ač tato verze protokolu nebyla nikdy oficiálně uvedena, podporoval ji například unixový klient PINE.

Počátkem 90-tých let se v IETF vytvořila pracovní skupina pro protokol IMAP a převzala zodpovědnost za verzi IMAP2bis. Bylo rozhodnuto o jeho přejmenování na IMAP4 z důvodu vyhnutí se kolize se zamýšleným standardem IMAP3, jež se ale nikdy neobjevil. IMAP se stal zkratkou Internet Message Access Protocol.

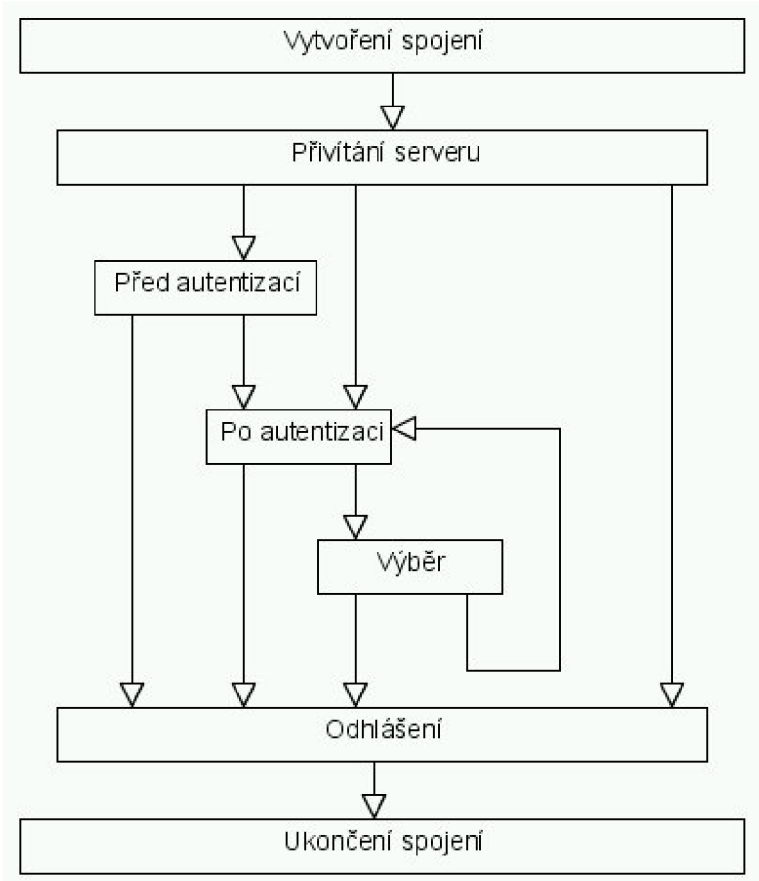
Některé chyby v původním IMAP4, jež byl definován v [8], které se objevily v průběhu implementací, vedly k revizi protokolu a k zatím poslední verzi IMAP4v1.

Současná verze z roku 1996 je definována v [10], která je revizí dřívější definice z [9]. IMAP4rev1 je zpětně kompatibilní z verzi IMAP2, IMAP2bis a samozřejmě také s IMAP4, ačkoliv starší verze již téměř vymizely.

4.2 Popis

Současná verze IMAP4 rev.1, jež byla definována v [10], umožňuje uživateli přistupovat a manipulovat s emaily přímo na serveru. Dále umožňuje přímou manipulaci se vzdálenými schránkami na serveru, jako by byly umístěny na lokálním počítači. IMAP4 dále také umožňuje synchronizaci pro offline klienty.

Ve standartu jsou definovány operace pro tvorbu, přejmenování a mazání emailových schránek, kontrolu zpráv, jejich mazání, vyhledávání v nich či nastavování příznaků zpráv. O každé zprávě si server uchovává seznam parametrů, kam patři např. různé příznaky (přečteno, odpovězeno apod.) či velikost zprávy. Bližší informace k tomuto protokolu jsou k nalezení v [10].



Obrázek 4.1: Návaznost stavů IMAP komunikace

4.3 Zabezpečení

Oproti ostatním starším definicím protokolů je v [10] definována možnost šifrované autentizace. Stále je ale také povoleno autentizace bez šifrování. Dále IMAP4 také umožňuje šifrování nejen samotné autentizace, ale i celé komunikace, a to buď použitím SSL nebo pomocí příkazu STARTTLS [14].

4.4 Komunikace

IMAP4 server standardně pracuje na portu 143. Relace IMAP4v1 sestává z vytvoření spojení, průvodního pozdravu od serveru a následné komunikace mezi klientem a serverem, jež obsahuje příkazy od klienta a data a odpovědi na příkazy od serveru. Komunikace mezi klientem a serverem probíhá ve formě textu ukončeného znakem konce řádku CRLF. Server může být ve čtyřech stavech v nichž lze použít jen některé příkazy. Návaznost jednotlivých stavů je uvedena na obrázku 4.1.

4.4.1 Stavy komunikací

- Před autentizací – klient je povinnen provést autentizaci, jinak mu nebude povoleno provádět většinu příkazů.

- Po autentizaci – v tomto stavu je klient povinnen zvolit schránku s kterou chce pracovat, jinak nebudou povoleny příkazy pro práci se zprávami. Tento stav může nastat po úspěšné autentizaci, po chybném výběru emailové schránky či po úspěšném provedeni příkazu uzavření zvolené schránky.
- Výběr – tento stav nastává po správném výběru schránky.
- Odhlášení – tento stav nastává po požadavku klienta na odhlášení příp. jako jednostranná akce klienta nebo serveru.

4.4.2 Odpovědi serveru

Odpovědi serveru se dělí na 3 druhy :

- Informace o statutu – jedná se o odpovědi na příkazy ve tvaru OK, NO, BAD, PREAUTH nebo BYE. Před OK, NO a BAD může být uveden kód příkazu s nímž daná odpověď souvisí. Odpovědi mohou dále obsahovat nepovinné kódy odpovědí.
- Data ze serveru – této odpovědi se používá pro přenos dat ze serveru na klienta. Server může zasílat např. informace o stavu jednotlivých zpráv, jejich seznam či obsah.
- Požadavky na další příkazy – tuto odpověď server používá, pokud od klienta očekává další informace, např. při autentizaci.

4.4.3 Příklad komunikace

Příklad komunikace mezi serverem a klientem je uveden na obrázku 4.2

4.5 Minimální požadavky na implementaci

Je vyžadována kompletní implementace dle [10].

```
S:   * OK IMAP4rev1 Service Ready
C:   a001 login mrc secret
S:   a001 OK LOGIN completed
C:   a002 select inbox
S:   * 18 EXISTS
S:   * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S:   * 2 RECENT
S:   * OK [UNSEEN 17] Message 17 is the first unseen message
S:   * OK [UIDVALIDITY 3857529045] UIDs valid
S:   a002 OK [READ-WRITE] SELECT completed
C:   a003 fetch 12 full
S:   * 12 FETCH (FLAGS (\Seen) INTERNALDATE "17-Jul-1996 02:44:25 -0700"
      RFC822.SIZE 4286 ENVELOPE ("Wed, 17 Jul 1996 02:23:25 -0700 (PDT)"
      "IMAP4rev1 WG mtg summary and minutes"
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      ((NIL NIL "imap" "cac.washington.edu"))
      ((NIL NIL "minutes" "CNRI.Reston.VA.US")
      ("John Klensin" NIL "KLENSIN" "MIT.EDU")) NIL NIL
      "<B27397-0100000@cac.washington.edu>")
       BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 3028
       92))
S:   a003 OK FETCH completed
C:   a004 fetch 12 body[header]
S:   * 12 FETCH (BODY[HEADER] {342}
S:   Date: Wed, 17 Jul 1996 02:23:25 -0700 (PDT)
S:   From: Terry Gray <gray@cac.washington.edu>
S:   Subject: IMAP4rev1 WG mtg summary and minutes
S:   To: imap@cac.washington.edu
S:   cc: minutes@CNRI.Reston.VA.US, John Klensin <KLENSIN@MIT.EDU>
S:   Message-Id: <B27397-0100000@cac.washington.edu>
S:   MIME-Version: 1.0
S:   Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
S:
S:   )
S:   a004 OK FETCH completed
C:   a005 store 12 +flags \deleted
S:   * 12 FETCH (FLAGS (\Seen \Deleted))
S:   a005 OK +FLAGS completed
C:   a006 logout
S:   * BYE IMAP4rev1 server terminating connection
S:   a006 OK LOGOUT completed
```

Obrázek 4.2: Příklad IMAP komunikace

Kapitola 5

Srovnání POP3 a IMAP

Při zpracování této sekce bylo využito [19].

5.1 Výhody POP3

- Složitost protokolu – vzhledem k POP3 je IMAP4 velmi komplikovaný protokol. Implementace IMAP4 je obtížnější a náchylnější na výskyt chyb jak pro server tak klienta, především z důvodu možnosti mnohonásobného přístupu ke schránce.
- Paměťová náročnost – přestože jsou algoritmy pro IMAP4 většinou velmi dobře implementovány, klient může spotřebovat velké množství zdrojů serveru, např. při prohledávání velkých emailových schránek.
- Rychlost připojení – s ohledem na architekturu IMAP4, musí klienti zprávu vždy přenášet ze serveru, což může způsobit zahlcení na pomalých linkách, např. při mobilním užití.

5.2 Výhody IMAP

- Velké maily – při použití POP3 se klient typicky připojuje k serveru jen pro stáhnutí zpráv, kdežto při použití IMAP4 si uživatel může vybrat které zprávy si chce číst, což v případě velkých emailů vede ke zlepšení reakční doby připojení.
- Víceuživatelský přístup ke schránce – POP3 vyžaduje exkluzivní přístup ke schránce, kdežto IMAP4 umožňuje víceuživatelský přístup a mechanismus pro detekci změn ostatními uživateli.
- Přístup k částem zpráv – téměř všechny maily jsou dnes přenášeny v MIME formátu, což znamená rozdělení zprávy na více částí. Protokol IMAP4 umožňuje stahovat jednotlivé části emailu bez nutnosti stahovat celou zprávu.
- Informace o stavu zpráv – IMAP4 umožňuje pomocí příznaků uživateli zaznamenávat si stav jednotlivých zpráv (přečtena, odpovězeno, přeposlaná apod.). Toto lze výhodně použít např. při použití více emailových klientu na více počítačích pro jednu schránku.
- Více schránek na serveru – IMAP4 klienti mohou vytvářet, přejmenovávat a/nebo mazat emailové schránky na serveru, jež jsou uživateli prezentovány jako složky, příp.

přesouvat jednotlivé maily mezi těmito schránkami. Vícesložkový přístup umožňuje používat některé složky jako soukromé či veřejné.

- Vyhledáváni na serveru – IMAP4 umožňuje vyhledávání ve zprávách na serveru, což nenutí uživatele je stahovat, a nezatěžuje tak procesor klienta.
- Možnosti rozšíření – s ohledem na zkušenosti s předchozími protokoly definuje IMAP4 explicitně způsob možného rozšíření. V současné době je v praxi již mnoho těchto rozšíření používáno.

5.3 Zhodnocení

S ohledem na uvedené výhody lze usoudit, že pro budoucí použití bude více podporován protokol IMAP, jež je sice náročnější na implementaci a konektivitu uživatele, ale zároveň mu poskytuje mnohem lepší kontrolu nad zprávami a mailovými schránkami.

Kapitola 6

SMTP

6.1 Popis

Počátky užívání SMTP protokolu jsou v raných 80-tých letech, kdy byl užíván jako doplněk k UUCP (Unix to Unix CoPy), jež byl vhodnější pro manipulaci s emaily mezi jednotlivými unixovými stroji, které nebyly neustále propojeny. Naproti tomu SMTP byl mnohem vhodnější, pokud odesílací a přijímací počítač byly neustále připojeny v sítí.

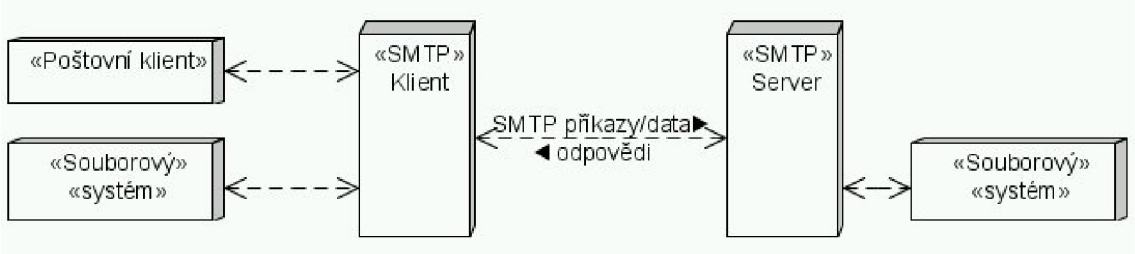
Jedním z prvních, ne-li první, server, který implementoval SMTP, byl SendMail, jež se používá dodnes. K dnešnímu dni již existuje velké množství klientů a serverů pro SMTP. Z klientů jmenujme například Mozilla Thunderbird, Microsoft Outlook, Pine, Elm čí Eudora E-mail. Ke známým SMTP serverům patří, kromě již zmiňovaného Sendmail, také Postfix, qmail, Novell NetMail či Microsoft Exchange.

SMTP je jednoduchý protokol, založený na komunikaci pomocí textových příkazů. Komunikační schéma je uvedeno na obrázku 6.1.

Jakmile chce klient odeslat zprávu, naváže spojení se svým SMTP serverem (na obrázku označen jako SMTP klient), kterému předá email k odeslání. SMTP klient následně kontaktuje SMTP server adresáta, kterému tento email předá.

SMTP na Internetu standardně užívá TCP port 25. Pro zjištění SMTP serveru adresáta se využívá systém DNS, z nějž se užívají MX (Mail eXchange) záznamy, případně A záznamy, pokud není MX záznam dostupný.

Tento protokol byl primárně určen pro přenos pouze textových emailů a neuměl si správně poradit s přenosem binárních dat. Postupným vývojem však tento stav přestal vyhovovat a tak bylo potřeba vyvinout standart jakým je MIME (Multipurpose Internet Mail Extensions), jež umožňuje zakódovat a přenést binární data přes SMTP. Bližší informace



Obrázek 6.1: Základní SMTP schéma

k protokolu jsou k nalezení v [12].

6.2 ESMTP

Jedná se o definici rozšíření protokolu SMTP, jež byla definována v [11]. Základní identifikační znak je pro ESMTP klienta použití příkazu EHLO při otevření spojení namísto HELO. ESMTP server by následně měl odpovědět potvrzovacím kódem „250 OK“ následovaným seznamem klíčových slov, které značí podporovaná rozšíření. Server nepodporující rozšíření ESMTP by měl odpovědět chybovým kódem a umožnit tak ESMTP klientovi přejít ke staršímu protokolu HELO případně ukončit spojení pomocí QUIT.

Každé rozšíření je definováno v samostatném RFC dokumentu a registrováno u organizace IANA. K nejpoužívanějším rozšířením patří :

- 8BITMIME – přenos 8-mi bitových dat
- AUTH – SMTP autentizace
- DSN – oznámení o stavu doručení
- HELP – pomocné informace
- PIPELINING – proudové zpracování příkazů
- SIZE – deklarace velikosti zprávy
- STARTTLS – zabezpečený přenos

Některé servery mohou používat také nestandardní a neregistrované rozšíření, které jsou v odpovědi na EHLO příkaz indikovány prefixem X před příkazem.

6.3 Zabezpečení

SMTP autentizace byla definována v [13] a poskytuje kontrolovaný přístup k serverům poskytujícím služby SMTP prtotokolu, čímž umožňuje legitimním uživatelům server užívat a nelegitimním, např. spammerům, je v přístupu zabráněno. Tato autentizace ale nezaručuje správnost údajů uvedených v emailu, takže tento mechanismus neřeší např. falšování adres odesílatele, kdy se jeden uživatel vydává ze někoho jiného.

Dále tento mechanismus umožňuje serverům si předávat informaci o tom, že uživatel se autentizoval, a umožnit tak předávání emailů mezi servery. Toto ale vyžaduje důvěru mezi jednotlivými servery, což je na dnešním Internetu problematické, a tudíž není tato možnost téměř vůbec využívána.

I přesto, že je SMTP autentizace bezpečnostním zlepšením oproti původnímu SMTP, nese v sobě jednu špatnou vlastnost. Pokud je autentizovaným uživatelům povoleno zasílat emaily z jakékoliv IP adresy, pak bezpečnost systému velmi silně závisí na bezpečnosti hesel, jež uživatelé užívají. Prolomením tohoto hesla je totiž umožněno zneužití serveru spammery pro rozesílání nevyžádané pošty.

6.4 Role serveru

K přenosu emailu obvykle dochází mezi SMTP serverem odesílatele a serverem SMTP adresáta. Může ale nastat situace, kdy tyto dva servery nejsou připojeny na stejnou síť. V takovém případě dochází k přenosu s pomocí přeposílacích serverů, případně skrze tzv. „gateway“, jež k dalšímu přenosu může používat jiný protokol než SMTP. Dle rolí lze tedy SMTP servery rozdělit na 4 skupiny :

- Zdrojový server – vkládá email na internet, případně jinou transportní službu.
- Cílový server – obdrží email z internetu a tento předá dále ke zpracování interním systémům k uložení, uživatel si jej následně vyzvedne ze své schránky.
- Přeposílací server – obdrží email a ten bez jakéhokoliv zpracování (kromě přidání informace a tom, že email prošel skrze tento server) přepošle dále přeposílacímu nebo cílovému serveru.
- Gateway server – obdrží email z jedné transportní služby a přepošle jej na jinou (např. TCP/IP na IPX/SPX). Rozdíly mezi jednotlivými protokoly, jež gateway užívá, mohou způsobit, že je potřeba provést převod formátu emailu. Jako gateway je možno brát například firewall, jež provádí NAT převod IP adres.

6.5 Komunikace

6.5.1 Stavy komunikace

- Iniciace relace – SMTP relace je vytvořena když klient otevře TCP spojení na servery a server odpoví otevírací zprávou. SMTP server může do své odpovědi vložit také informace identifikující verzi software, jež používá. Tohoto se dá použít k efektivnějšímu určení a následné opravě problémů. Zároveň to ale může vést k bezpečnostním rizikům, proto ke doporučeno umožnit zákaz této identifikace. SMTP protokol dále umožňuje serveru odmítnout transakci, i když je povoleno vytvořit připojení na server.
- Iniciace klienta – po odeslání uvítací zprávy serverem a přijmutí této zprávy klientem, tento odešle EHLO příkaz serveru. Toto je znamením pro server, že klient podporuje rozšířenou funkčnost a požaduje po serveru seznam podporovaných rozšíření. Starší klienti, kteří toto rozšíření nepodporují, mohou využít starší komunikační schéma a příkaz HELO . Zároveň také v případě, že klient zašle příkaz EHLO a server mu odpoví „command not recognized“ (příkaz nerozpoznán), musí být klient schopen komunikace s pomocí HELO.
- Mail transakce – vkládání emailu na SMTP server probíhá ve 3 krocích. Transakce začíná příkazem MAIL, který obsahuje identifikaci odesílatele. Následně je pomocí příkazu RCPT vložen jeden či více příjemců a pomocí příkazu DATA je zahájen přenos dat emailu, jež je ukončen zasláním samostatné tečky na řádku.
- Ukončení spojení – SMTP spojení je ukončeno, když klient zašle příkaz QUIT. Server odpoví kladnou odpovědí a následně uzavře spojení. Bližší informace jsou k nalezení v [12]

6.5.2 Odpovědi od serveru

Server odpovídá na každý příkaz odpovědí. Tyto odpovědi mohou znamenat přijetí příkazu, čekání na další příkazy příp. že nastala dočasná či permanentní chyba. Konstrukce těchto odpovědních kódu je následující :

- 1xx – Předběžná pozitivní odpověď – příkaz byl přijat, ale požadovaná akce byla odložena a server čeká na potvrzení případně zrušení akce od klienta.
- 2xx – Kompletní pozitivní odpověď – požadovaná akce byla provedena a je očekáván nový požadavek.
- 3xx – Částečná pozitivní odpověď – příkaz byl přijat, požadovaná akce byla odložena a server čeká na další informace pro upřesnění. SMTP klient by měl zaslat další příkaz k specifikující tyto informace. Tato odpověď je používána při sekvenčních příkazech, např. DATA.
- 4xx – Přechodně negativní odpověď – příkaz nebyl přijat a požadovaná akce nebyla provedena. Vzniklá chyba ale není permanentní a klient může zaslat nezměněný příkaz znovu a ten bude proveden.
- 5xx – Trvalá negativní odpověď – příkaz nebyl přijat a požadovaná akce nebyla provedena. Tato chyba je trvalá.

Druhá číslice rozděluje odpovědi do kategorií následovně :

- x0x – syntaxe příkazu
- x1x – informace
- x2x – spojení
- x5x – mailový systém

Třetí číslice zpřesňuje definici chyby určené druhou číslicí.

Například na příkaz NOOP, jehož provedení nedává klientovi žádné nové informace a pouze brání vypršení spojení, odpoví server odpovědí 250. Pokud není příkaz implementován, následuje odpověď 502, případně 504, pokud je příkaz implementován, ale je požadován např. neimplementovaný parametr.

6.5.3 Příklad komunikace

Příklad komunikace je uveden na obrázku 6.2. Další příklady komunikací jsou k nalezení např. v [12].

6.6 Další funkcionalita

- Přeposílání emailu – Tato funkcionalita je nejčastěji potřebná pro sjednocení a zjednodušení adres v organizacích, případně k přesměrování zpráv ze staré adresy uživatele na jeho novou. K tomuto přesměrování může dojít tajně, bez informování uživatele, případně s informováním uživatele, kdy ale systém nesmí předpokládat, že si uživatel změnu v adrese zaznamenal a musí tuto informaci zveřejňovat dále.

```
S: server
C: klient

S: 220 foo.com Simple Mail Transfer Service Ready
C: EHLO bar.com
S: 250-foo.com greets bar.com
S: 250-8BITMIME
S: 250-SIZE
S: 250-DSN
S: 250 HELP
C: MAIL FROM:<Smith@bar.com>
S: 250 OK
C: RCPT TO:<Jones@foo.com>
S: 250 OK
C: RCPT TO:<Green@foo.com>
S: 550 No such user here
C: RCPT TO:<Brown@foo.com>

S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: Blah blah blah...
C: ...etc. etc. etc.
C: .
S: 250 OK
C: QUIT
S: 221 foo.com Service closing transmission channel
```

Obrázek 6.2: Příklad SMTP komunikace

- Ověřování adres – SMTP umožňuje také provádět ověřování správnosti adres, příp. získávat obsahy mailových seznamů, tzv. mailing listů. Tato funkčnost je podporována skrze příkazy VRFY a EXPN, přičemž není striktně vyžadována a s ohledem na bezpečnost může být na systémech, které ji podporují, záměrně zakázána.

- Explicitní směrování – Explicitní směrování je technika používaná v historii pro určení trasy, po níž má daný email procházet. Bližší informace jsou k nalezení v [17]. Její užití je však díky MX záznamům v DNS zcela zbytečné a s ohledem na mnoho historických problémů, jež tato technika zapříčinila, je její užití nežádoucí. SMTP klienti by tedy neměli tuto techniku používat a SMTP servery mohou odmítnout tento email zpracovat, případně tyto směrovací informace ignorovat a kontaktovat SMTP server uvedený jako poslední.

6.7 Minimální požadavky na implementaci

Pro minimální implementaci jsou požadovány následující příkazy :

- EHLO, HELO, MAIL, RCPT, DATA, RSET, NOOP, QUIT, VRFY

- Každý SMTP server podporující doručování a přeposílání pošty musí podporovat speciální adresu „postmaster“. SMTP systémy by měl zaručit doručitelnost na tuto adresu za všech okolností. Popis příkazů je k nalezení v [12].

Kapitola 7

Služby ve Windows NT

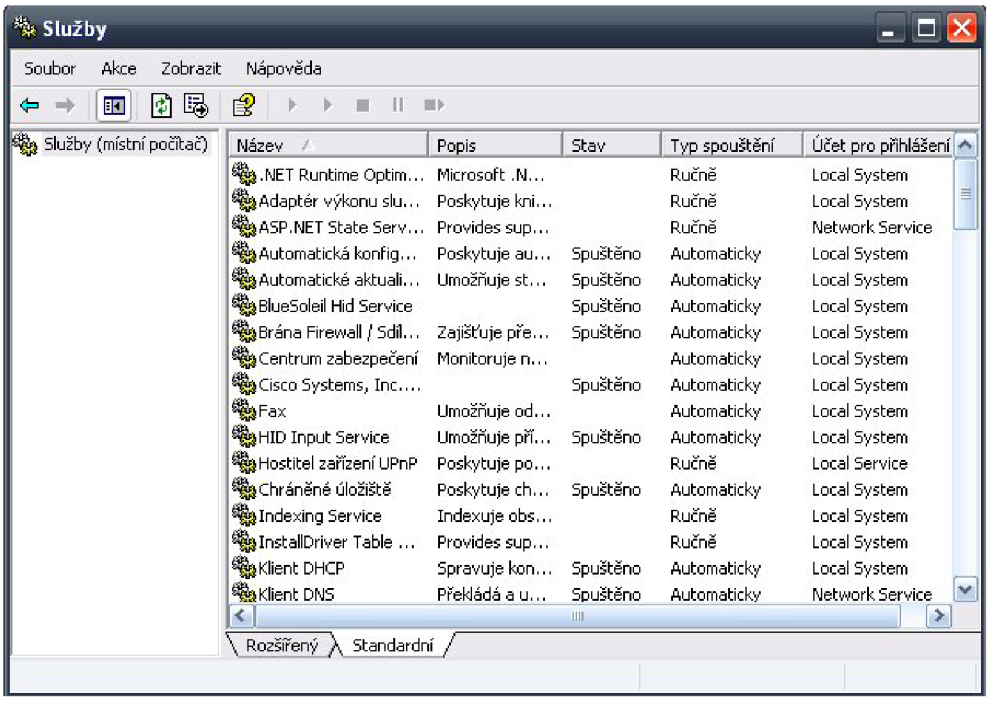
7.1 Základní popis

V operačních systémech řady Windows NT, tj. Windows NT, 2000, XP, 2003 Server a Vista, existuje možnost užívání tzv. services (služeb). Jedná se o program, jenž běží od začátku spuštění operačního systému a to bez nutnosti přihlášení uživatele. Toto je první odlišnost od standardních aplikací. Jako služby běží například sdílení souborů, jednoduché TCP/IP služby či sdílení internetového připojení. Většina služeb běží pod systémovým účtem, což znamená možnost provádět na daném počítači téměř jakékoliv akce. Běží v pozadí, většinou bez možnosti interakce s uživatelem, veškerá komunikace probíhá skrze logování událostí do logů.

- Výhody služeb
  - K jejich spuštění není potřeba přihlášení uživatele.
  - Běží vždy, nezávisle na přihlášeném uživateli.
  - Mají dostatečná oprávnění pro provádění jakýchkoliv operací.
  - Mohou být nakonfigurovány pro běh ihned po spuštění a příp. restart, pokud je ukončena, tzn. mohou běžet po celou dobu běhu systému.
- Nevýhody služeb
  - Bezpečnostním rizikem je jejich běh pod nejvyššími právy. Jakákoliv chyba v implementaci může znamenat bezpečnostní ohrožení systému.
  - Protože běží bez komunikace s uživatelem, uživatel obvykle neví, že služba běží.
  - Ne všichni správci znají popis všech služby a tak často běží služby, které nemusí, jen kvůli neznalosti správce a zbytečně spotřebovávají systémové prostředky.

Protože služby běží mimo jakýkoliv uživatelský kontext, je v operačních systémech řady NT (dále OS) proces na jejich správu, který se jmenuje Service Control Manager (SCM) viz. obr. 7.1. Ten ví o všech službách nainstalovaných na daném OS a zná jejich stav (zda běží či nikoliv). Stará se zároveň o automatické spuštění služeb po startu OS a o požadavky ze strany uživatelů na spuštění či zastavení služby.

Když je OS spuštěn, SCM zkontroluje, které služby mají být po startu automaticky spuštěny a spustí je. Jak bylo dříve zmíněno, služba je spuštěna standardně pod systémovým účtem, ale v OS existuje možnost nastavení spuštění pod jakýmkoliv jiným lokálním účtem



Obrázek 7.1: Správce služeb ve Windows XP CZ

daného OS, což může zmírnit bezpečnostní rizika. Správce systému má tedy možnost vytvořit speciální účet pro každou službu a definovat jí jistá omezení, pokud je to nutné. K tomuto kroku se ale většina správců neuchyluje, protože jen správa těchto účtů a jejich správné nastavení je velmi náročné na čas a znalosti.

Jednodušším a často používanějším přístupem, jsou omezení, kdy měnit stav služby může pouze správce systému a službě je zakázán přístup k významným souborům, např. „cmd.exe“. Takto lze omezit rizika plynoucí z použití služeb a využití jejich plného potenciálu.

7.2 Životní cyklus

Služba prochází během své „doby života“ několika stavy. Nejdříve je služba nainstalována na systém, na němž má běžet. Tzn. je spuštěn instalátor služby, který ji nahraje do SCM.

Po nahrání čeká služba na své spuštění. Po spuštění začne provádět akce, pro něž byla vytvořena.

Běžící služba může v tomto stavu setrvat tak dlouho, dokud není ukončena, pozastavena čí OS vypnut.

Kapitola 8

Návrh architektury serveru

Cílem této kapitoly je seznámení s prostředím, v němž bude výsledná aplikace navržena, a návrhem jednotlivých částí, jež budou později implementovány.

8.1 Rozvržení aplikace

Základním předpokladem pro správnou funkčnost je vhodné rozvržení aplikace. Už při seznámení s jednotlivými protokoly, jež musí emailový server implementovat, je zcela zřejmé základní rozdělení na moduly implementující protokoly a pomocný modul, jež bude poskytovat funkce pro správu dat na disku.

- Modul SMTP serveru – modul bude obstarávat minimální požadovanou funkcionalitu nutnou pro správnou funkci SMTP serveru dle [**?**]. Bude očekávat spojení na požadovaném portu (standardně 25), přijímat maily a ty následně předávat modulu pro doručování.
- Modul pro doručení mailů(MTA) – modul bude ukládat maily do lokálních složek uživatelů příp. je předávat na jiné servery, kde má uživatel svou schránku. Jeho součástí bude také jednoduchý DNS klient, jež bude sloužit ke komunikaci s DNS servery a bude poskytovat modulu informace o požadovaných doménách, především o MX záznamech, jež jsou potřebné pro zjištění správné adresy SMTP serveru adresáta.
- Modul POP3 serveru – úkolem modulu bude obstarávat minimální požadovanou funkcionalitu nutnou pro správnou funkci POP3 serveru dle [3]. Bude naslouchat na daném portu (standardně 110) a obsluhovat veškerou komunikaci s klienty.
- Modul IMAP serveru – modul bude implementovat verzi protokolu IMAP4rev1 dle [10]. Bude naslouchat na požadovaném portu (standardně 143) a obsluhovat komunikaci s klienty.
- Modul správy souborů(MMA) – modul bude mít na starost veškerou manipulaci se soubory obsahující mailové zprávy, vytváření uživatelských účtů, složek na disku a veškerou další funkcionalitu nutnou pro spolupráci se souborovým systémem.

8.2 Ukládání zpráv

Pro ukládání zpráv na emailovém serveru existuje obrovské množství možností. Většina těchto možností využívá k ukládání zpráv souborový systém případně databázi. Pro tento server byl zvolen systém ukládání v souborovém systému, kdy bude existovat základní adresář serveru, jež bude obsahovat základní uživatelské složky s podsložkami.

8.2.1 Zprávy v souborovém systému

Zprávy v souborovém systému budou ukládány jako jednotlivé soubory. Jejich jméno se bude skládat ze 4 částí. První část bude obsahovat unikátní identifikátor zprávy v rámci dané složky, v druhé bude obsaženo datum doručení zprávy na server. Třetí část bude obsahovat seznam příznaků zprávy a poslední částí bude připona zprávy „*.email“.

Následující seznam příznaků bude možno nastavit pro každou zprávu:

- FLAGGED – volný příznak pro použití klienta.
- ANSWERED – na zprávu bylo odpovězeno.
- SEEN – zpráva byla přečtena.
- DELETED – zpráva byla přesunuta do koše.
- DRAFT – jedná se o návrh zprávy.
- RECENT – nově příchozí zpráva.

8.2.2 Doručení zprávy

Doručení zprávy provádí modul pro doručení mailů společně s modulem správy souborů. Tato akce probíhá v několika krocích :

- Modul MTA převezme od SMTP modulu přijatou zprávu
- Modul MTA rozdělí adresáty na lokální a vzdálené.
- MTA předá MMA modulu zprávy pro lokální uživatele a ten je uloží do jejich složek a označí jako nově příchozí.
- MTA získá adresu vzdáleného SMTP serveru adresáta a pokusí se na něj připojit a předat zprávu k doručení.

Pokud modul MTA selže při doručování zpráv, musí o tom podat zprávu odesílateli zprávy.

8.2.3 Čtení zprávy

Při čtení zpráv MMA modul získá seznam všech zpráv v požadované složce a tento předá modulu POP3/IMAP serveru. Poté je možno se zprávou libovolně manipulovat v rámci použitého protokolu. Pokud zpráva zůstane po manipulaci s ní na serveru, je opět předána zpět MMA modulu, který provede požadované operace a změny uloží.

8.3 Konfigurace

Pro hlavní konfiguraci serveru bude sloužit jeden XML soubor, který bude obsahovat veškeré informace nutné pro běh. Zároveň bude existovat seznam uživatelů, jež bude obsahovat hesla pro přístup do schránky a pak pro každou složku soubor, který bude obsahovat informace o složce. Hlavní konfigurační soubor serveru bude modifikovatelný pomocí lokálního rozhraní, které poběží na serveru.

Kapitola 9

Implementace

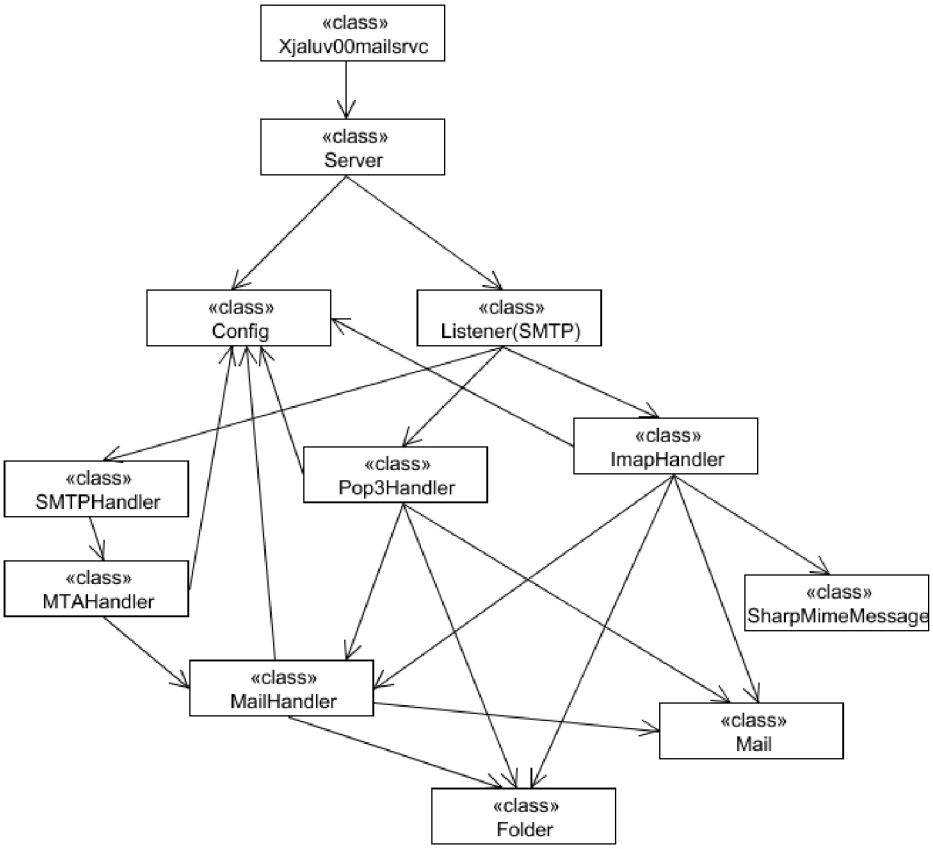
Před samotnou implementaci bylo potřeba zvolit vhodné prostředí a jazyk, v němž bude server vytvořen. Na operačních systémech společnosti Microsoft existuje široké spektrum nástrojů pro vývoj aplikací v různých jazycích. S ohledem na předchozí zkušenosti s vývojem jiného software a požadavkem použití jako služba na systému s technologií Windows-NT, byly zavrženy možnosti použití prostředí Windows API v kombinaci s jazyky C/C++ a následně také Javy.

Nevýhodou použití Windows API je nutnost provádět veškerou obsluhu systémových volání ručně, což by vedlo k velmi složitému kódu a tak znamenalo možnost vzniku velkého počtu chyb(s ohledem na komplikovanost této obsluhy). Existuje sice možnost použití knihoven třetích stran, ale ty, dle dosavadních zkušeností, mohou mít problémy při použití na různých verzích operačních systému Microsoft.

Další možností bylo použití jazyku Java. Ten oproti WinAPI poskytuje nástroje pro jednoduchou implementaci a obsluhu systémových služeb, ale, dle posledních vyjádření, společnost Microsoft ukončila vývoj a distribuci vlastního virtuálního stroje, jež je nutný pro běh aplikací v tomto jazyce a tak by se mohl opět vyskytnout problém s kompatibilitou příp. nepodporováním některých funkcí.

Jako nejvhodnější se proto pro implementaci jeví prostředí .Net Framework, které je vyvíjeno samotnou společností Microsoft, čímž je zaručena kompatibilita se všemi podporovanými operačními systémy této společnosti. Toto prostředí je velmi podobné dříve zmiňovanému prostředí jazyku Java, kdy poskytuje programátorovi prostředky pro jednoduchou obsluhu systémových služeb a tak zásadně přispívá k rychlejšímu vývoji a sňižuje riziko výskytu chyb. Pro implementaci bylo zvoleno toto prostředí v aktuální verzi 2.0. Podobně jako při překladu v jazyce Java dochází k transformaci do mezikódu, zde nazývaného MSIL(Microsoft Independent Language), díky čemuž je např. možno v jednom projektu kombinovat několik různých jazyků.

Pro implementaci v tomto prostředí byl zvolen jazyk, jež byl společností Microsoft vyvinut právě pro prostředí .Net, jazyk C#. Jeho výhodou oproti C++, jež je možno v prostředí .Net také použít, je např. neexistence ukazatelů, pouze referencí, větší typová bezpečnost díky omezení automatických konverzí na minimum a jednodušší správa paměti, díky automatickému uvolňování pamětí pomoci garbage collectoru. Tím si je C# naopak podobný s jazykem Java, s kterým má i další podobné prvky jako např. neexistenci násobné dědičnosti, jež je nahrazena použitím rozhraní(interface).



Obrázek 9.1: Třídy serveru a jejich provázanost

9.1 Princip funkce serveru

Na obr. 9.1 je znázorněn model serveru a provázanost jednotlivých tříd. Základní třídou je *Xjaluv00MailSrvc*, což je základní třída umožňující běh serveru jako služba. Tato třída zabezpečuje provádění metod, jež jsou implementovány ve třídě *Server*. Jedná se o spouštění a zastavování serveru. Při spuštění serveru třída *Server* načte konfiguraci a vytvoří 3 instance(pro každý protokol jednu) třídy emphListener, jež čekají na příchozí spojení. Jakmile k tomuto dojde, je vytvořena, v závislosti na typu protokolu, odpovídající instance třídy pro obsluhu protokolu *Pop3/Imap/SMTPHandler*, jež se dále stará o obsluhu tohoto spojení a komunikuje s klientem dle daného protokolu, což, v případě protokolu SMTP, znamená autentizaci uživatele(pokud je vyžadována), přijetí mailu a jeho předání vytvořené instanci třídy *MTAHandler*, jež se postará o jeho doručení. V případě, že se jedná o POP3 či IMAP, tato komunikace znamená výběr mailů ze složky a manipulace s nimi, v závislosti na příkazech klienta.

9.2 Možnosti nastavení

Pro dobrou funkčnost serveru, je potřeba mít uloženo několik základních údajů, jako jsou porty, na nichž má očekávat příchozí spojení, kde jsou umístěna uživatelská data, které ob-

sluhy protokolů spustit apod. K tomuto uložení existuje několik možností. K nejzajímavějším možnostem, kde mohou být údaje uloženy patří:

- registry systému
- databáze
- souborový systém

V sekci návrhu bylo definováno, že pro ukládání konfiguračních údajů bude sloužit XML soubor, z čehož vyplývá nevhodnost systémových registrů jako úložiště. Jednou z vlastností serveru by měla být jeho jednoduchost pro provoz a správu. Proti této podmínce by zřejmě bylo použití databázového serveru pro ukládání konfiguračních dat, protože by se správa mailového serveru ještě rozrostla o správu a konfiguraci databázového systému, což by již nemělo s jednoduchostí moc společného. Pro ukládání konfigurace byl tedy zvolen souborový systém.

Pro ukládání konfigurace byla vytvořena třída *Config*, která sdružuje veškeré nastavení serveru a obsahuje funkce pro ukládání a nahrávání konfigurace. Konfigurace je vždy načtena při spuštení služby, změny je možno provádět ručně v souboru, který se nachází v systémovém adresáři (např. *c:\windows\system32*) pod názvem *xjaluv00mail.xml*.

Obecná nastavení

- BaseDir – absolutní cesta k adresáři, ve kterém jsou uloženy uživatelské schránky
- CertPath – absolutní cesta k certifikátu nutného pro spuštění TLS (Transport Layer Security)

Nastavení SMTP

- SmtpRun – informace pro program, zda spustit SMTP server
- SmtpPort – číslo portu, na kterém ma běžet SMTP server
- SmtpMaxMsgSize – maximální velikost odesílaných dat v bytech
- SmtpMaxRecipients – maximální počet příjemců mailu
- SmtpTimeout – doba bez aktivity, po které je spojení považováno za neaktivní
- SmtpAuthReq – zda se musí uživatel před pokusem o odeslání mailu autentizovat

Nastavení MTA

- MtaMaxRetries – maximální počet pokusů o odeslání, pokud dojde k chybě při odesílání
- MtaRetryInterval – doba v minutách mezi jednotlivými pokusy o odeslání, pokud dojde k chybě
- MtaUseSmartHost – zda použít pro odeslání zpráv vzdálených uživatelů jeden SMTP server
- MtaHost – adresa SMTP serveru pro odesílání zpráv vzdáleným uživatelům

- MtaPort – port, na kterém očekává SMTP server připojení
- MtaAuth – zda použít pří odesílání autentizaci
- MtaUser – uživatelské jméno pro přístup na vydálený SMTP server
- MtaPass – heslo pro přístup na vydálený SMTP server

Nastavení IMAP

- ImapRun – informace pro program, zda spustit IMAP server
- ImapPort – číslo portu, na kterém má běžet IMAP server

Nastavení POP3

- Pop3Run – informace pro program, zda spustit POP3 server
- Pop3Port – číslo portu, na kterém má běžet POP3 server

Načtení konfigurace

Konfigurace je v programu prezentována instancí třídy *Config*, na disku pak xml souborem. Při startu služby je potřeba data z tohoto xml souboru načíst do instance třídy *Config*. K tomuto načtení se používá proces zvaný (de)serializace, jež je podporován přímo .Net Frameworkem. Výhodou xml (de)serializace pro ukládání a přenos objektů je především její jednoduchost díky textové prezentaci dat a samopopisnému formátu, jež vyplýva z definice formátu xml. Pro xml (de)serializaci platí následující pravidla:

- ukládají se pouze vlastnosti objektů a veřejně přístupné proměnné(modifikátor public)
- neukládá se informace o typu třídy
- aby byla třída serializovatelná, musí deklarace obsahovat defaultní konstruktor
- pro všechny vlastnosti, které se mají serializovat, musí existovat možnost čtení i zápisu

Všechny tyto podmínky jsou v třídě *Config* splněny. K (de)serializaci je v této třídě použit standartní objekt .Net Frameworku *XmlSerializer*.

9.3 Uživatelé a schránky

Organizace

Pro evidenci uživatelů je možno zvolit, podobně jako při ukládání konfigurace, z několika možností. S ohledem na jednoduchost konfigurace a následné správy, byl zvolen přístup, kdy je namísto externího seznamu uživatelů, který by bylo potřeba neustále udržovat synchronizovaný s aktuálním stavem, použito jako uživatelský účet jméno existující složky v základním adresáři mailového serveru, která obsahuje jako podsložky všechny složky, které si uživatel vytvoří pomocí funkcí protokolu IMAP, a také soubor *mailbox.conf*, jež obsahuje hash hesla pro přístup do dané schránky, vytvořeného pomocí algoritmu SHA-512, v base64 kódování. Vzhledem k použití protokolu IMAP je potřeba si o složkách a v nich uložených zprávách uchovávat informace o unikátním identifikátoru dané složky,

unikátním identifikátoru zprávy a jestli je daná složka označena jako odebíraná(subscribed) protokolem IMAP pro daného uživatele. K tomuto slouží soubor *folder.conf*, který obsahuje všechny tyto údaje v textové podobě. Jako unikátní ídentifikátor složky je zvoleno datum vytvoření v binárním formátu. Pro unikátní identifikaci zpráv je použita rostoucí posloupnost přirozených čísel a jako příznak odebírání dané složky je použit textový řetězec true/false.

Manipulace se schránkami a účty

K vytváření a editaci účtů byl vytvořen program *Admin*, který v aktuální verzi implementuje pouze jednoduché textové rozhraní pro tvorbu uživatelských účtů(schránek) a změnu hesla. Veškerá další manipulace se schránkami ze strany uživatele by měla probíhat pomocí protokolu IMAP. Mazání uživatelských účtů a (pod)složek je možno případně provádět jednoduchým smazáním dané složky v souborovém systému. Vzhledem k zaměření vyvíjeného serveru k lokálnímu provozu, kdy má uživatel možnost přístupu do své složky pomocí síťových služeb jako jsou SAMBA příp. FTP, byl vývoj prioritně zaměřen na dokončení implementace protokolů, před tvorbou rozhraní pro vzdálený přístup a správu.

9.4 Přístup k souborovému systému

Pro spolupráci mezi souborovým systémem, který obsahuje uložená data v podobě mailových zpráv a uživatelských složek, byla vytvořena třída *MailHandler*, která implementuje funkcionalitu nutnou pro manipulaci se zprávami, autentizaci uživatelů a výlučný přístup do složek.

9.4.1 Manipulace se zprávami

Třída *Mail*

O každé zprávě, s níž je na serveru manipulováno, je potřeba mít informace, které jsou uloženy na disku ve jménu souboru, který danou zprávu obsahuje. K získání těchto dat existují 2 přístupy. Při každém přístupu k dané zprávě načíst jméno souboru z disku a z něj následně oddělit požadované údaje nebo mít tyto údaje načteny v paměti systému a provést rozdělení údajů pouze jednou, při prvním přístupu ke zprávě klientem. Výhodou prvního přístupu je aktuálnost načítaných dat, nevýhodou naopak nízká rychlost z důvodu neustálého provádění oddělování textových polí. Druhý přístup naopak trpí neaktuálností načtených dat(pokud mezitím došlo k manipulaci s danou zprávou např. jiným připojeným klientem), ale poskytuje mnohem vyšší rychlost přístupu k údajům. S ohledem na nutnou rychlost přístupu k dané zprávě, byl tedy zvolen druhý přístup.

K uchovávání těchto údajů o zprávě je použita třída Mail, která obsahuje následující údaje:

- unikátní identifikátor zprávy
- interní datum, kdy došlo k uložení zprávy na server
- seznam příznaků zprávy
- velikost zprávy

Součástí této třídy jsou také metody pro nastavování příznaků dané zprávy, která je interně reprezentována bitovým polem.

Třída *Folder*

Při předávání načtených informací o zprávách ve složce mezi jednotlivými třídami serveru, musí byt tyto zprávy uspořádány v nějaké struktuře. S ohledem na podmínky protokolů POP a IMAP pro přístup zpráv dle indexu, byla jako vhodná struktura zvoleno pole. Zároveň je také potřeba o složce uchovávat informace, které jsou uloženy v souboru *folder.conf*, a dále např. relativní cestu ke složce v uživatelském adresáři.

K uchovávání údajů o složce a obsažených zprávách je použita třída *Folder*, která obsahuje následující údaje:

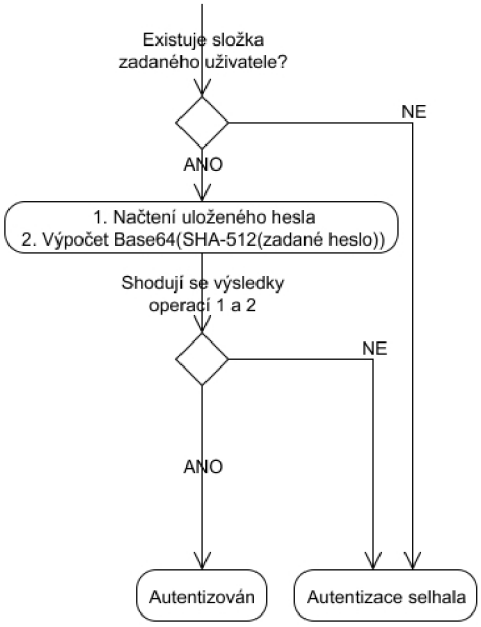
- relativní cestu ke složce vzhledem k uživatelskému adresáři
- absolutní cestu ke složce v souborovém systému
- unikátní identifikátor složky
- příznak, zda může být složka vybrána klientem
- příznak, zda klient odebírá obsah této složky (IMAP - subscribed)

Dále jsou součástí třídy také metody pro zjištění a aktualizaci následujícího unikátního čísla příchozí zprávy ve složce.

Požadované operace pro práci se zprávami

K zajištění korektní funkčnosti serveru tato třída implementuje následující typy operací, jež pokrývají veškerou aktuálně nutnou funkcionalitu pro práci se zprávami/složkami:

- Uložení dat zprávy – při přijetí zprávy je potřeba provést režijní úkony jako je zajištění uložení zprávy do správné složky, zjištění unikátního identifikátoru nové zprávy spolu s aktualizací počítadla, uložení času přijetí a správného nastavení příznaků nové zprávy. Tuto režii má na starost metoda *SaveMsg()*.
- Načtení dat zprávy – po připojení klienta je potřeba poskytnout mu data, která jsou ve zprávě uložena. Ty nejsou umístěna spolu s údaji v instanci třídy Mail, protože v některých případech, kdy zpráva obsahuje velké množství dat v řádu MB, by mohlo lehce dojít k vyčerpání přidělené paměti systému. Ke zpřístupnění těchto dat je implementována metoda *getData()*, jež data z disku načte a vrátí v textovém řetězci.
- Synchronizace informací o zprávě – po manipulaci se zprávou pomocí podporovaných protokolů může dojít ke změně příznaků. Zpráva může být označena jako přečtená, přeposlaná případně označena ke smazání. Promítnutí těchto změn, jež jsou uloženy v paměti, do souborového systému zajišťují metody *DeleteMails(), RemoveFile(), UpdateMails() a UpdateFlags()*.
- Správa složek – při připojení klienta tento vyžaduje seznam všech zpráv na serveru v dané složce. K tomuto načtení dat ze souborového systému slouží metoda *getFolder()*. Dále pak protokol IMAP obsahuje příkazy, které vyžadují implementaci metody pro načtení seznamu podsložek v zadané složce. Tato funkcionalita je implementována metodou *listFolders()*. Protokol IMAP pak také obsahuje další příkazy pro manipulaci se jmény složek a jejich obsahem, tyto však nejsou v současné verzi implemetovány.



Obrázek 9.2: Postup autentizace uživatele

9.4.2 Autentizace uživatelů

Při připojení klienta k serveru je nutné provést jeho autentizaci. Jak již bylo dříve zmíněno, k evidenci uživatelů není použit žádný soubor, ve kterém by byl seznam všech uživatelů serveru, ale samotné složky v základním adresáři serveru. Každá tato složka obsahuje soubor s hashem, jež je výsledkem aplikace algoritmu SHA-512 na uživatelské heslo, v kódování base64. Díky uložení hashe namísto hesla v nezakódované podobě, není prakticky možné zjistit uživatelovo heslo, což přispívá k větší bezpečnosti uživatelova účtu, pokud by došlo ke zcizení daného souboru a pokud používá toto heslo i v jiných systémemch. Zároveň ale také není možná jednoduchá autentizace pouhým porovnáním uloženého řetězce s příchozím heslem, zaslaným klientem. Z toho důvodu třída *MailHandler* obsahuje metody pro převod příchozího hesla v nezakódované podobě na odpovídající formu(*HashPass()*) a jeho porovnání s uloženým heslem(*AuthUser()*) - postup autentizace je znázorněn na obr. 9.2. Pro změnu hesla je pak k dispozici metoda *chPass()* a především pro použití MTA modulu, implemetuje tato třída metody *isLocal()*, které slouží k ověření, odpovídá-li zadané uživatelské jméno příp. mailová adresa lokálnímu uživateli.

9.4.3 Výlučný přístup do složek

Protokol POP3 vyžaduje pro přístup do složky příchozí pošty zaručení exkluzivního přístupu. K zaručení tohoto typu přístupu existuje v .Net Frameworku mutex(zkratka z anglického mutually exclusive (access)). Jedná se o vzájemné vyloučení přístupu paralelně běžících úkolů k jednomu jedinému společnému zdroji, zde složce příchozí pošty. Paralelními úkoly mohou být různé úkoly (vlákna) jak jedné aplikace (jednoho procesu), tak jiné aplikace. V jednu chvíli může mutex vlastnit nanejvýše jeden úkol. Toto je zaručeno použitím me-

tody třídy *MailHandler acquireLock()*, která se pokusí vytvořit mutex pro exkluzivní přístup do zadané složky.

9.5 Obsluha připojení

9.5.1 Akceptace připojení

Pro zahájení komunikace mezi klientem a serverem, je potřeba, aby systém očekával příchozí připojení od klienta a následně jej předal vhodnému procesu, který bude schopen komunikovat s klientem dle požadovaného protokolu.

Server je na síti identifikován svou IP adresou, která samotná však k rozlišení jaký protokol použít ke komunikaci s klientem nepostačuje. K tomuto odlišení se používají porty. Hodnota portu může nabývat hodnot 0 až 65535, přičemž hodnoty 0 až 1023 se používájí především pro standardní služby a jejich správu má na starost společnost IANA. Seznam aktuálně přidělených portů lze nalézt na jejích stránkách *www.iana.org*. Pro implementované protokoly jsou dle IANA rezervovány porty 25 pro SMTP, 110 pro POP3 a 143 pro IMAP. Pro zabezpečené verze těchto protokolů, které však nejsou tímto serverem aktuálně podporovány, jsou pak rezervovány porty 995 pro POP3s a 993 pro IMAPs, pro zabezpečené připojení na SMTP se většinou používá port 465. Ten je však u organizace IANA registrován firmou Cisco pro URD, proto je pro zabezpečení SMTP doporučováno použití TLS společně v kombinaci s portem 587, viz. [2].

O příjem příchozích spojení a jejich předání správné třídě se stará třída *Listener*. Server při svém startu spustí pro každý povolený podporovaný protokol jednu instanci této třídy, jež naslouchá na všech síťových rozhraních na požadovaném portu a příchozím spojení pak předá správné třídě k další komunikaci dle protokolu.

9.5.2 Komunikace

Po přijetí připojení třídou *Listener* a jeho předání odpovídající třídě je potřeba načítat příkazy zaslané klientem, tyto zpracovat a odeslat jako odpověď odpovídající data. Ke komunikaci je možno zvolit 2 druhy přístupu (obr. 9.3):

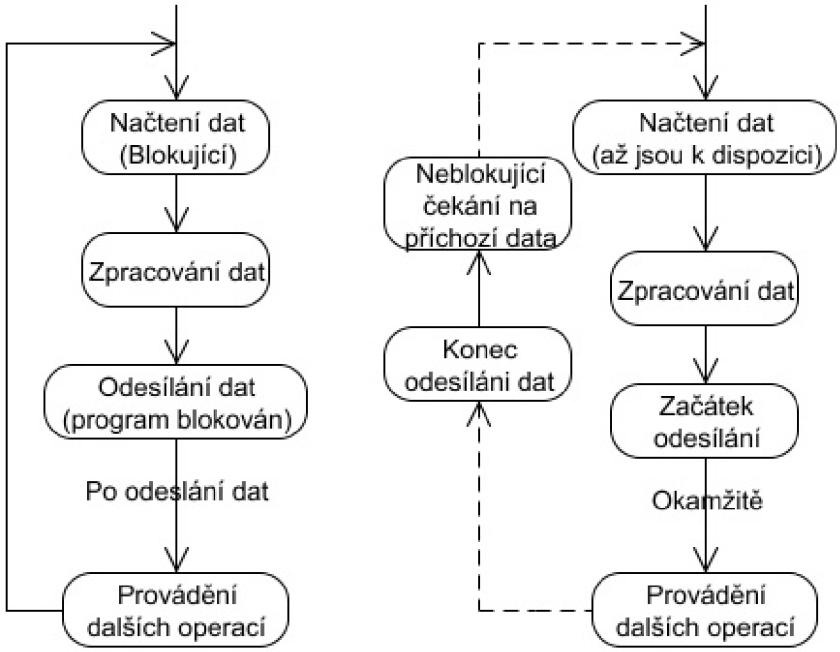
- Synchronní – operace odesílání/načítání čekají, dokud neproběhne odeslání/načtení dat, pak je možno pokračovat v dalším běhu programu
- Asynchronní – operace jsou rozděleny na 2 části, začátek odesílání/příjmu a konec odesílání/příjmu. Nedochází tak k blokování programu a je možno pokračovat v jeho provádění bez čekání na odeslání.

Vzhledem na nedostatek synchronního přístupu, jímž je blokace programu při odesílání a čtení dat, což by mohlo znamenat např. zbytečné zdržení aktualizace zpráv na disku a následné užití již neaktuálních dat, bylo zvoleno k obsluze komunikace mezi klientem a serverem použití asynchronních operací.

9.6 Doručování zpráv

9.6.1 Modul SMTP serveru

Pro příjem zpráv od klientů je použita třída *SMTPHandler*, která implementuje serverovou část protokolu SMTP.



Obrázek 9.3: Příklad synchronní(vlevo) a asynchronní(vpravo) komunikace

Zpracování příkazů

SMTP je textovým protokolem a tak veškerá komunikace probíhá pomocí předem definovaných textových příkazů dle rfc 2821[12]. Zároveň je SMTP stavovým protokolem, což znamená, že některé příkazy jsou platné pouze v určitém stavu. Proto je pro implementaci funkce pro zpracování příkazů nejvhodnější využití stavového automatu. Stavový automat pro zpracování příkazů funguje následujícím způsobem:

1. Převod dat na textovou reprezentaci – protokol SMTP je textovým protokolem, ale pro komunikaci na transportní vrstvě se pracuje s poli bajtů. Před operacemi s příkazem je tedy potřeba provést konverzi dat na textovou formu.

2. Provedení kontroly na speciální stavy – data obdržené od klienta většinou začínají příkazem SMTP protokolu. Toto ale neplatí ve stavech, kdy dochází k autentizaci, nebo jsou přijímána data. Pokud se jedná o některou z těchto možností, jsou data zpracována zde a na základě výsledku zpracování je odeslána odpověď.

3. Rozdělení dat na příkaz a parametry – SMTP příkaz se obvykle skládá ze samotného příkazu a jeho parametrů, jako je např. emailová adresa. Pro jednodušší následné zpracování SMTP příkazu je nutné provést toto rozdělení.

4. Rozpoznání SMTP příkazu – v tomto bodě je určena metoda, která provede zpracování příkazu.

5. Provedení příkazu – metoda provede zpracování zaslaných parametrů a příp. změní aktuální stav automatu.

6. Zaslání odpovědi – jako reakce na zaslaný příkaz je, dle výsledku provedení požadované akce, zaslána odpověď klientovi - viz. odpovědi od serveru v kapitole o protokolu SMTP.

Uložení zpráv k odeslání

Pro příjem zprávy, kterou chce klient odeslat a předá ji pomocí SMTP protokolu serveru, se používá datový typ string(řetězec). Výhodou tohoto typu v prostředí .Net je automatická alokace paměti a z toho vyplývající jeho téměř neomezená velikost. Přijatou zprávu je však potřeba předat MTA modulu, jež má na starost její doručení adresátovi a tak by mohlo dojít ke stavu, kdy se zprávu nepodaří doručit na první pokus a v průběhu doby, než dojde k dalšímu pokusu o doručení, je přijato několik dalších větších zpráv, které způsobí nedostatek paměti systému. Proto je potřeba se tomuto přístupu, kdy jsou v systému načteny všechny zprávy k doručení, vyhnout použitím dočasného souboru a k předávání zpráv použít pouze odkaz na tento dočasný soubor.

9.6.2 Modul pro doručení mailů - MTA

Po doručení zprávy klientem serveru je potřeba zprávu uložit do složek lokálních uživatelů příp. předat dále SMTP serveru adresáta. Tuto funkcionalitu implementuje třídá *MTAHandler*, zkráceně MTA(Mail Transfer Agent). Oproti třídě *SMTPHandler*, jež implementuje serverovou část protokolu SMTP, tato obsahuje klientskou část protokolu.

Doručování zpráv

V závislosti na nastavení serveru, jsou zprávy doručovány dvěma způsoby. Pokud není nastaveno použití SMTP přeposílacího serveru, tzv. smart hostu, dochází k odesílání zprávy přesně, jak bylo popsáno v sekci návrhu.

1. Třída *SMTPHandler* předá spolu se seznamem adresátu odkaz na dočasný soubor, jež obsahuje zprávu k odeslání.
2. Rozdělení seznamu adresátů na lokální a vzdálené
3. Uložení zprávy do lokálních schránek
4. Získání MX záznamů z DNS systému pro vzdálené uživatele
5. Připojení na SMTP servery vzdálených uživatelů a předání zprávy k doručení – pokud se tento krok nezdaří, server jej několikrát s přestávkami opakuje(v závislosti na nastavení) a pokud dojde k vyčerpání povolených pokusů, je lokální uživatel informován chybovou zprávou.

V případě, že správce serveru nastaví použití přeposílacího serveru k doručení zpráv skrze tento server, což může být provedeno například z důvody firemní síťové politiky, liší se postup doručení tím, že namísto zjištění adres SMTP serverů adresátů, se server pokusí získat adresu SMTP serveru pro nastavený přeposílací počítač a pak mu předá zprávy pro všechny vzdálené adresáty.

Komunikace s DNS

I přesto, že je prostředí .Net Framework velmi obsáhlé a pokrývá velkou část potřebných systémových služeb, neexistuje v něm jednoduchý způsob, jak získat z DNS serverů MX záznam, který obsahuje informace o adrese SMTP serveru pro hledanou doménu. Proto je potřeba implementovat tuto metodu pomocí vlastních sil a naimporotvat požadované funkce pro komunikaci z externích DnsAPI knihoven operačního systému. Implementované metody pro tyto operace, použité v tomto serveru, jsou převzaty od Petera Bromberga a jejich přesný popis a způsob užití je k nalezení na [16].

9.7 Čtení zpráv

Pro čtení a manipulaci se zprávami mail server podporuje 2 protokoly. Jedná se o protokol POP3, jehož obsluhu implementuje třída *Pop3Handler*, a protokol IMAP4v1, jehož obsluhu implementuje třída *ImapHandler*.

9.7.1 POP3

Princip funkce

POP3 je, podobně jako SMTP, textovým protokolem a tak veškerá komunikace mezi klientem a serverem probíhá pomocí definovaných textových příkazů. POP3 je rovněž stavovým protokolem a tak je jeho obsluha, podobně jako pro SMTP, prováděna konstrukcí založenou na stavovém automatu. Schéma běhu automatu je znázorněno na obr. 9.4. Jedná se o pohled na vyšší úrovní abstrakce, detailní popis je dostupný ve formě zdrojového kódu, jež je k této práci připojen.

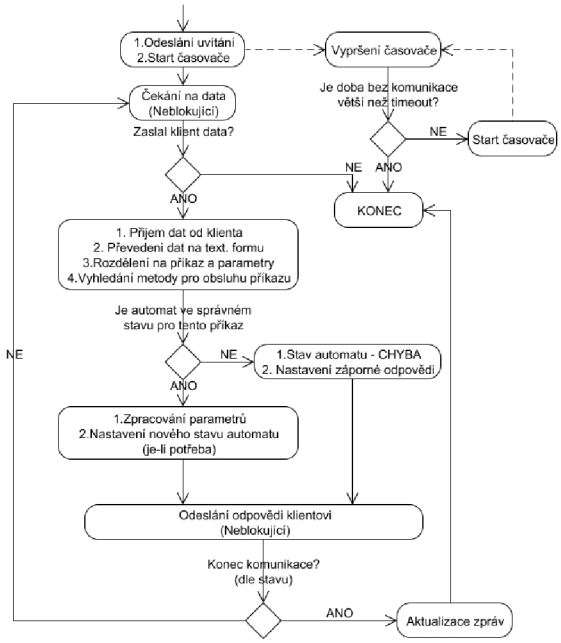
Nové stavy spojení

Protokol POP3 definuje několik stavů v nichž se může spojení nacházet. Jedná se o tyto stavy:

- Autorizace – před zasláním uživatelského jména a hesla
- Transakce – po zaslání jména a hesla a výběru složky pro příchozí poštu
- Aktualizace – po odhlášení klienta od serveru

Uvedené stavy však nejsou pro správnou funkci serveru dostačující. Je potřeba ještě dále odlišit další 2 stavy, které se mohou vyskytnout a je v nich potřeba specifického chování. Proto byla definice stavů spojení rozšířena o tyto 2 nové stavy:

- Částečná autorizace – při zasílání hesla neobsahují klientem zasílaná data žadné příznaky, dle nichž by se dalo rozpoznat, jestli tomuto předcházelo zaslání uživatelského jména. Z tohoto důvodu bylo potřeba zavést tento nový interní stav, do nějž spojení přechází po zaslání uživatelského jména.
- Ukončení – při uzavření spojení s klientem přechází toto do stavu aktualizace. Je však potřeba rozlišit, došlo-li jen k uzavření spojení z důvodu chyby při komunikaci nebo zda-li se uživatel úspěšně odhlásil. Proto byl zaveden tento nový stav, do nějž stav spojení přechází po nekorektním ukončení spojení, např. zasláním příkazu ve stavu, v němž není povolen.



Obrázek 9.4: Schéma práce třídy *Pop3Handler*

Bezpečnost

Na Internetu se bohužel nevyskytují pouze poctiví uživatelé, kteří se chovají dle standardů a nepokoušejí se někoho podvést či využít ku vlastnímu prospěchu. Vyskytují se zde hackeří, snažící se prolomit hesla uživatelů, spammeři, jejichž obživa spočívá v zasílání různých nevyžádaných mailů do co největšího počtu scránek. Je proto potřeba poskytnout jistou úroveň bezpečnosti i při užití pop3 protokolu. Není sice v silách serveru odolat např. útoku hrubou silou, ale již je v jeho možnostech pokusit se tento útok alespoň významně zpomalit. Proto se implementace tohoto serveru drží zásady neposkytovat v odpovědích více, než jen nutné informace, jež jsou potřeba pro korektní funkci protokolu.

Další bezpečnostním prvkem je princip zacházení s operací pro přihlášení uživatele - USER. Některé servery se neřídí doporučeními a při zadání neznámého uživatelského jména ihned oznámí, že daný uživatelský účet na serveru neexistuje. Tato vlastnost je sice výhodná v tom, že klient se o zadání špatného uživatelského jména dozví ihned po jeho zadání a nemusí tak zadávat zbytečně heslo, ale v případě, že se jedná o útočníka, mu tímto vstřícným krokem dáme vědět, že daný uživatelský účet na tomto serveru není a může zkoušet zadat jiné uživatelské jméno příp. se přesunout na další server, kde se o přihlášení pod tímto účtem pokusí opět. Implementace tohoto serveru se doporučením, jež se týká podávání informací o existenci či neexistenci zadaného účtu, řídí. Uživateli tedy není žádným způsobem signalizováno, že zadal neexistující uživatelské jméno a je vyzván k zadání hesla pro přístup do scránky. Po jeho zadání je pak pouze informován o zadání chybného hesla, z čehož nemůže usoudit, zda na serveru zadaný uživatel má či nemá účet.

9.7.2 IMAP4v1

Protokol IMAP je oproti protokolu POP3 velmi komplexním protokolem a jeho implementace je tak mnohem obtížnější a programově významně rozsáhlejší. Proto byl při implementaci třídy *ImapHandler* zvolen postup, kdy byl kladen důraz na implementování základní funkcionality, nutné pro připojení klienta a následný přístup ke složce doručené pošty, namísto postupné implementace jednotlivých příkazů.

Princip funkce

Princip funkce IMAP serveru je velmi podobný POP3 serveru, jež je znázorněn na obr. 9.4, proto již zde nebude opětovně uveden, ale budou zdůrazněny pouze významné odlišnosti.

- Předzpracování dat – před samotným rozdělením dat na části obsahující příkaz a jeho parametry, je potřeba provést předzpracování. Toto předzpracování je nutné z důvodu mírné odlišnosti ve specifikaci oproti protokolu POP3 pro zasílání dat klientem, kdy klient zašle příkaz a pomocí speciálního ukončení příkazu serveru signalizuje, že očekává jeho souhlas k zaslání doplňujících údajů. Pokud k tomuto stavu dojde(tento postup používají někteří klientu např. pro zasílání přihlašovacích údajů), je potřeba definovaným způsobem informovat klienta o svolení k zaslání dodatečných dat. Dokud tato data nejsou klientem zaslána, není možno pokračovat ve zpracování příkazu.

- Sekvenční zasílání příkazů – protokol IMAP povoluje, oproti protokolu POP3, zasílání několika příkazů současně. Proto je potřeba provádět zpracování dat zaslaných klientem po řádcích a tím se vyhnout chybnému zpracování obdržených dat, kdy by byl identifikován příkaz a veškerá dále došlá data od klienta jako jeho parametry. Bližší informace k tomu, o které příkazy se jedná jsou k nalezení v [10]

- Aktualizace zpráv – IMAP byl navržen jako protokol pro manipulaci se zprávami na serveru a ne jako protokol pro jejich pouhé stahování ze serveru. Proto obsahuje, oproti POP3, i příkazy pro explicitní aktualizaci uložených zpráv, tudíž k aktualizaci zpráv nedochází pouze při odpojení klienta, což by bylo vzhledem k délce trvání IMAP spojení nežádoucí(s ohledem na aktuálnost dat), ale i explicitně na žádost klienta.

- Ukončení spojení – v případě, že IMAP server nerozpozná klientem zaslaný příkaz nebo tento příkaz není v aktuálním stavu spojení povolen, nedochází k přechodu serveru do chybového stavu a k uzavření spojení(jako v případě POP3 serveru), ale server zůstane ve stavu, v němž se nacházel před přijetím daného příkazu a o chybě klienta pouze informuje.

Zpracování MIME zpráv

Protože je protokol IMAP navržen pro práci se zprávami uloženými na serveru a umožňuje stahování pouze segmentů zpráv, jako jsou např. vybraná pole z hlavičky příp. pouze některé přílohy, musí umět dekódovat a zpracovat zprávy ve formátu MIME. K tomuto dekódování je použito volně šiřitelné knihovny *SharpMimeTools* ve verzi 0.6, jež je šířena pod licencí GNU GPL. Bližší informace jsou k nalezení na *http://anmar.eu.org/projects/sharpmimetools*.

9.8 Běh jako služba

Tvorba a následné ladění služeb pro systémy založené na technologii Windows NT bylo ještě v nedávné době velmi komplikovanou a pracnou záležitostí. S příchodem .Net Frameworku však tento problém mizí, protože pro tvorbu služeb, nově nazývaných Windows Services(díky odchodu systémů nezaložených na technologii NT do záhrobí), poskytuje .Net základní bázové třídy, které maji na starost veškerou správu a komunikaci se systémem, kterou musel dříve provádět programátor ručně, a tak odpadá řada problémů. Zároveň prostředí .Net poskytuje utilitu pro instalaci takto vytvořených služeb, program installutil.exe, takže nic nebrání v jejich instalaci a použití.

Instalace/odinstalace služby

K instalaci služby vytvořené v prostředí .Net Framework je určena aplikace installutil.exe, která se nachází obvykle v adresáři, v němž je nainstalováno prostředí .Net framework, např. *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727*.

- INSTALACE
  - installutil.exe „absolutni_cesta_k_programu_serveru“
- ODINSTALACE
  - installutil.exe -u „absolutni_cesta_k_programu_serveru“

Spuštění služby

Ke spouštění a kontrole služeb v systémech řady Windows-NT slouží správce služeb. Po instalaci dle výše uvedeného příkladu se služba nainstaluje jako služba spouštěna manuálně a proto je jí potřeba ručně spustit příp. zastavit ve správci služeb.

Kapitola 10

Možnosti použití a dalšího rozšíření

10.1 Aktuální funkčnost

V aktuální verzi je plně podporován protokol SMTP a POP3. Implementace protokolu IMAP není úplná. Jak bylo zmíněné v sekci implementace protokolu IMAP, byl při implementaci zvolen přístup kladoucí důraz na funkčnost serveru pro příjem pošty. Pro vývoj byli proto zvoleni 3 typičtí zástupci mailových klientů, dle nichž byla implementace vedena. Jednalo se o následující klienty:

- Mozilla Thunderbird CZ 1.5.0.10
- Opera 9.20 build 8771
- Outlook Express v.6.00.2900.2180

Aktuální verze tedy dovoluje, s použitím těchto verzí klientů, bezproblémový příjem pošty. S jinými verzemi a klienty pro příjem pošty implementovaný server nebyl testován a nelze proto zaručit jeho funkčnost.

Pro plnou podporu protokolu IMAP je nutné doimplementovat následující příkazy: STARTTLS, EXAMINE, CREATE, DELETE, RENAME, SUBSCRIBE, UNSUBSCRIBE, APPEND, CLOSE, SEARCH a COPY.

10.2 Možnosti rozšíření

Existuje velké množství možností pro úpravy serveru, ať už na uživatelské či systémové úrovni. V této části jsou navrženy a diskutovány některé z nich.

WWW rozhraní pro přístup do schránky

Ne vždy má uživatel možnost přistoupit do své schránky použitím vlastního poštovního programu a když se právě vyskytuje u počítače, jež je připojen do internetu a mailový server je tak dostupný, většinou mu není, z bezpečnostní důvodů, povoleno nastavení poštovního klienta dle jeho potřeb, příp. uživatel není schopen nebo ochoten se prokousávat nastavením klienta pro odesílání či přijímání pošty. Proto se jako velmi vhodné rozšíření jeví použití webové rozhraní pro přístup do schránky.

Pro implementaci tohoto rozhraní je možno zvolit 2 přístupy:

- Přizpůsobení serveru některé již hotové implementaci – tato možnost je vyhodná z hlediska časových nároků na implementaci, protože je potřeba obvykle pouze provést rozšíření implementovaného protokolu o některá nová rozšíření, a většinou i větší bezpečnosti a menšímu počtu chyb, protože dané řešení je otestováno na velkém počtu uživatelů(toto platí i v případě málo rozšířené implementace, kde se jedná o jednotky až desítky nasazení v provozu).
- Implementace vlastní aplikace – tato možnost obsahuje větší možnosti kontroly nad směrem vývoje dané aplikace, jednodušší uzpůsobení vlastním potřebám. Jedná se však o časově velmi náročnou možnost, z důvodu potřeby vlastního testování a vývoje většinou v malém týmu.

Zásuvné moduly

Zasuvné moduly jsou soubory, díky jejichž použití se dá rozšířit funkčnost aplikace a které zprostředkovávají spolupráci mezi serverem a externími programy. Pro umožnění vývoje těchto zásuvných modulu je potřeba navrhnout rozhraní, které umožní jejich použití a zvolit jejich vhodné umístění v hierarchii serveru. Jako nejvhodnější místo pro jejich použití se jeví místo vstupu zprávy do systému, těsně před uložením do uživatelovy schránky, kdy je možné provést např.:

- virovou kontrolu obsahu zprávy
- kontrolu, zda se jedná o nevyžádanou zprávu
- aplikaci filtrovací funkce, na jejímž výsledku dojde ke správnému zatřídění zprávy do požadované složky

Kapitola 11

Závěr

E-mail je sice jednou z nejstarších, ale zároveň také jednou z nejpoužívanějších služeb na Internetu. Každý uživatel může, díky free mailovým serverům, vlastnit velké množství mailových adres. Pro vytvoření a správu vlastního jednoduchého mailového serveru na platformě Windows-NT však téměř neexistují volně dostupná řešení. V této zprávě byly čtenáři představeny základní protokoly používané pro emailovou komunikaci. Čtenář se seznámil s jejich výhodami a nevýhodami a následně také s návrhem architektury jednoduchého mailového serveru pro platformu Windows-NT. Následně byl navržený server implementován a čtenář byl seznámen s jeho architekturou, postupem při implementaci a pak také s jejími významnými částmi. Implementaci protokolu IMAP se s ohledem na komplexnost tohoto protokolu a jeho rozsáhlost nepodařilo včas dokončit. Potvrdilo se tak, že implementace IMAP protokolu je nejen časově, ale i finančně velmi náročná a z toho důvodu se na trhu téměř nevyskytují volně dostupná řešení pro platformu Windows-NT. I přes nekompletní podporu protokolu IMAP je však zaručena funkčnost s nejpoužívanějšími autorovi známými poštovními klienty. V 10.kapitole pak byly diskutovány možnosti dalšího vývoje a možného rozšíření serveru.

Literatura

[1] J. Myers a Carnegie Mellon. RFC 1734 - POP3 AUTHentication command [online], December 1994 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc1734.txt.

[2] R. Gellens a J. Klensin. RFC 2476 - Message Submission [online], December 1998 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc2476.txt.

[3] J. Myers a kolektiv. RFC 1939 - Post Office Protocol - Version 3 [online], May 1996 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc1939.txt.

[4] L. Lundblade a kolektiv. RFC 2449 - POP3 Extension Mechanism [online], November 1998 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc2449.txt.

[5] N. Freed a N. Borenstein. RFC 2045-2049 - Multipurpose Internet Mail Extensions [online], November 1996 [cit. 29.12.2006]. Dostupné z WWW:
http://www.ietf.org/rfc/rfc2045.txt,
http://www.ietf.org/rfc/rfc2046.txt,
http://www.ietf.org/rfc/rfc2047.txt,
http://www.ietf.org/rfc/rfc2048.txt,
http://www.ietf.org/rfc/rfc2049.txt.

[6] M. Crispin. RFC 1064 - INTERACTIVE MAIL ACCESS PROTOCOL - VERSION 2 [online], July 1988 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc1064.txt.

[7] M. Crispin. RFC 1176 - INTERACTIVE MAIL ACCESS PROTOCOL - VERSION 2 [online], August 1990 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc1176.txt.

[8] M. Crispin. RFC 1730 - INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4 [online], December 1994 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc1730.txt.

[9] M. Crispin. RFC 2060 - INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1 [online], December 1996 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc2060.txt.

[10] M. Crispin. RFC 3501 - INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1 [online], March 2003 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc3501.txt.

[11] J. Klensin a kolektiv. RFC 1869 - SMTP Service Extensions [online], November 1995 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc1869.txt.

[12] J. Klensin. RFC 2821 - Simple Mail Transfer Protocol [online], April 2001 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc2821.txt.

[13] J. Myers. RFC 2554 - SMTP Service Extension for Authentication [online], March 1999 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc2554.txt.

[14] C. Newman. RFC 2595 - Using TLS with IMAP, POP3 and ACAP [online], June 1999 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc2595.txt.

[15] Ian Peter. The history of email [online], 2004 [cit. 29.12.2006]. Dostupné z WWW: http://www.nethistory.info/History of the Internet/email.html.

[16] Ph.D. Peter A. Bromberg. Build a C# DNS MX (Mail Exchange) Record Query Class [online], 29.01.2005 [cit. 30.3.2007]. Dostupné z WWW: http://www.eggheadcafe.com/articles/20050129.asp.

[17] Jonathan B. Postel. RFC 821 - SIMPLE MAIL TRANSFER PROTOCOL [online], August 1982 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc821.txt.

[18] P. Resnick. RFC 2822 - Internet Message Format [online], April 2001 [cit. 29.12.2006]. Dostupné z WWW: http://www.ietf.org/rfc/rfc2822.txt.

[19] WWW. Wikipedia. http://www.wikipedia.org/.

Dodatek A

Uživatelská dokumentace

A.1 Instalace

A.1.1 Požadavky

Server je implementován na platformě .Net Framework 2.0. Před instalací je tedy potřeba mít toto prostředí nainstalováno, nejlépe v aktuální verzi, minimálně však ve verzi 2.0. Instalační program pro aktuální verzi 3.0, jejíž jedinou odlišností oproti předchozí verzi 2.0 je přítomnost nových komponent pro Windows Vista, lze nalézt na adrese *http://www.microsoft.com/downloads/details.aspx?FamilyId=10CC340B-F857-4A14-83F5-25634C3BF043*.

A.1.2 Instalace služby

Server se skládá z jediného souboru, *MailServer.exe*. K instalaci služby do systému je pro platformu .Net Framework poskytována aplikace *installutil.exe*, která je umístěna v místě instalace .Net Frameworku. Obvykle se jedná o adresář systému Windows, např. *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727*. Program serveru umístíme do zvolené složky na disku a instalaci provedeme spuštěním *installutil.exe* s cestou k programu serveru jako parametr. Odinstalaci lze provést stejným programem, pouze je nutno před cestu k souboru serveru uvést přepínač *-u*. Pokud program serveru umístíme např. do složky *C:\Windows*, pak pro instalaci bude sloužit příkaz *installutil c:\Windows\MailServer.ex*, pro odinstalaci pak *installutil -u c:\Windows\MailServer.exe*. Průběh instalace je zachycen na obr. A.1.

A.1.3 Podporovaní klienti

V aktuální verzi, jsou podporováni následující klienti:

- Mozilla Thunderbird CZ 1.5.0.10
- Opera 9.20 build 8771
- Outlook Express v.6.00.2900.2180

A.2 Spuštění

Po instalaci je server nastaven ve správci služeb jako služba spouštěná uživatelem, jak je vidět na obr. A.2. Je ji tedy potřeba spustit, případně nastavit, aby byla spouštěna automaticky. Po tomto spuštění se v systémovém adresáři vytvoří základní konfigurační soubor

Obrázek A.1: Instalace serveru

xjaluv00mail.xml a základní složka serveru *xjaluv00mail* v kořenovém adresáři systémového disku.

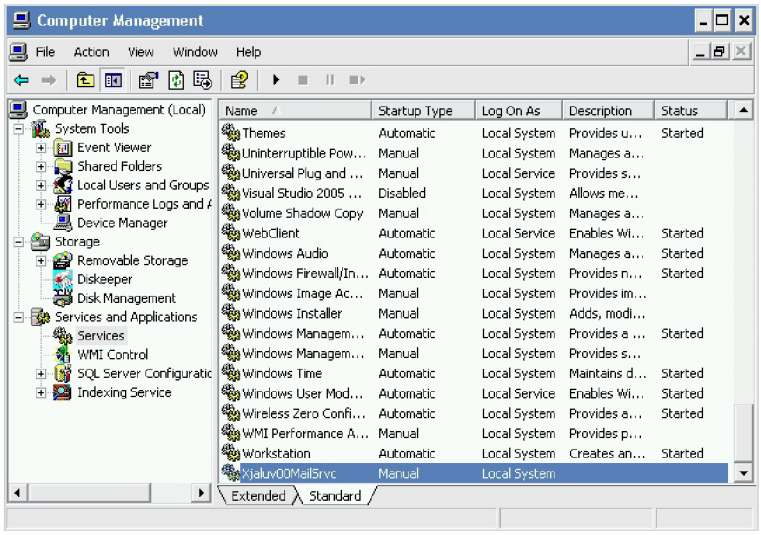
A.3 Konfigurace

A.3.1 Vlastnosti serveru

Ke konfiguraci serveru slouží xml konfigurační soubor *xjaluv00mail.xml*, jež se nachází v systémovém adresáři, např. *C:\Windows\system32*. Tento soubor je vytvořen při prvním spuštění serveru, příp. níže popsaného nástroje pro správu účtů. Obsah takto vytvořeného souboru je zobrazen na obr. A.3, popis jednotlivých parametrů je uveden v kapitole 9.2.

A.3.2 Správa účtů

Ke správě účtů slouží program *admin.exe*. Jedná se o jednoduchý konzolový program s textovým rozhraním, jež umožňuje tvorbu uživatelských účtů a případně změnu jejich hesel pro přístup. Další manipulace, jako mazání účtů, či jednotlivých složek, je možná pomocí standardních nástrojů pro práci se soubory.



Obrázek A.2: Stav služby po instalaci

```
<?xml version="1.0" encoding="utf-8"?>
<MailServer.Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <BaseDir>C:\xjaluv00mail</BaseDir>
  <CertPath />
  <SmtpRun>true</SmtpRun>
  <SmtpPort>25</SmtpPort>
  <SmtpMaxMsgSize>26214400</SmtpMaxMsgSize>
  <SmtpMaxRecipients>100</SmtpMaxRecipients>
  <SmtpTimeout>600</SmtpTimeout>
  <SmtpAuthReq>true</SmtpAuthReq>
  <MtaMaxRetries>3</MtaMaxRetries>
  <MtaRetryInterval>10</MtaRetryInterval>
  <MtaUseSmartHost>false</MtaUseSmartHost>
  <MtaHost />
  <MtaPort>0</MtaPort>
  <MtaAuth>false</MtaAuth>
  <MtaUser />
  <MtaPass />
  <Pop3Run>true</Pop3Run>
  <Pop3Port>110</Pop3Port>
  <ImapRun>true</ImapRun>
  <ImapPort>143</ImapPort>
</MailServer.Config>
```

Obrázek A.3: Základní nastavení konfiguračního souboru