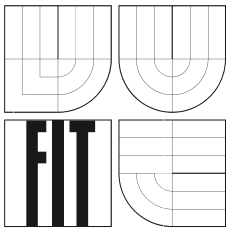
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Přátelské posílání dat

Ročníkový projekt

2006 **Jiří Chocholáč**

Přátelské posílání dat

Odevzdáno na Fakultě informačních technologií Vysokého učení technického v Brně
dne 2. května 2006

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Aleše Smrčky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Jiří Chocholáč
2. května 2006

Abstrakt

Tento projekt se zabývá návrhem a implementací nástroje pro posílání souborů mezi uživateli. Posílání souborů znamená připojit se k nějakému serveru a tam soubory nahrát a oznámit příjemci, na jakém serveru jsou soubory k dispozici, nebo spojit se přímo s počítačem příjemce a soubory mu nahrát. Projekt tento problém řeší pomocí dvou aplikací. První alikace slouží k připojení se k zadanému FTP serveru a uložení zásilky souborů. Poté aplikace odešle příjemcům zásilky souborů informaci o tom, kde si zásilku může vyzvednout. Druhá aplikace je vlastně webové rozhraní, kde po zadání kódu odpovídajícího dané zásilce může uživatel přenášet soubory v zásilce na svůj počítač.

Klíčová slova

přenos souborů, FTP, HTTP, e-mail, klient, server

Poděkování

Děkuji vedoucímu tohoto projektu Ing. Aleši Smrčkovi za poskytnuté rady a připomínky k této práci a také za trpělivost. Děkuji přátelům a rodině za podporu. Bez nich by práce nejspíše nevznikla.

Abstract

This project handles design and implementation of the tool designated to send files between users. Sending files means connect to the server, save files there and inform receiver about arrival. Receiver gets an information about the server, where the files are around. The project is trying to solve this problem with two aplications. First aplication must connect to the FTP server and save package of files there. Afterwards an aplication send an e-mail to the receivers of package notice information. Second aplication is web interface, with which receiver is able to reach his files in the package and save them on his computer.

Keywords

transfer files, FTP, HTTP, e-mail, client, server

Obsah

Kapitola 1

Úvod

Snad každý člověk, který někdy pracoval s počítačem, se setkal s potřebou přenesení souborů, případně celých adresářů, na jiný počítač. Běžného uživatele by pravděpodobně napadlo poslat takový soubor v příloze e-mailu. Toto řešení je samozřejmě nejjednodušší, pokud se jedná o jeden či dva soubory přiměřené velikosti. Pro přemístění objemnějších zásilek, jako jsou například soubory s multimediálním obsahem, je však toto řešení nepoužitelné vzhledem k serverem omezené velikosti posílaných e-mailů. Laik by v takovém případě buď rezignoval na přenos po síti, vypálil soubory na CD a dopravil jej k adresátovi osobně či poštou, a nebo se obrátil na odborníka.

Odborník by znal několik dalších variant řešení. Nahrát soubory na webové stránky a oznámit adresátovi, kde si je může stáhnout. Nevýhodou je, že by si soubory mohl okopírovat každý návštěvník stránek. Další možností by bylo uložení souborů na FTP server. Pro úspěšné dokončení přenosu by však bylo nutné prozradit adresátovi své uživatelské jméno a heslo, nehledě k tomu, že běžný uživatel obvykle účet na FTP serveru nevlastní. Nevýhodou přenosu prostředníctvím peer2peer sítě nebo pomocí Instant messangeru by byla nutnost současného připojení odesilatele i adresáta.

Shrnutím těchto poznatků překvapivě dospíváme k závěru, že problém jednoduchého přenosu dat konkrétní osobě nebyl dosud uspokojivě vyřešen. Účelem tohoto projektu je poskytnout běžnému uživateli, ale i odborníkovi nástroj, který by toto posílání souborů bez velkých omezení umožnil.

Kapitola 2

Charakteristiky používaných protokolů

V této kapitole, zejména pak při popisu jednotlivých protokolů používám některé základní pojmy s tímto významem:

- **protokol** - soubor pravidel, podle kterých dvě entity (například běžící programy) komunikují.
- **server** - obecně je to označení pro proces nebo systém, který poskytuje nějakou službu.
- **klient** - označení pro proces, nebo systém, který službu serveru využívá.
- **RFC** - zkratka anglického výrazu **request for comments** (žádost o komentáře), která se používá pro označení řady standardů a dalších dokumentů popisujících Internetové protokoly. RFC jsou oficiálně považovány spíše za doporučení než normy v tradičním smyslu, ale přesto se podle nich řídí drtivá většina Internetu.
- **URL** - zkratka anglického výrazu **Uniform Resource Locator**. Je to řetězec znaků s definovanou struktourou sloužící k přesné specifikaci zdrojů informací (dokumentů, nebo služeb) na Internetu. například URL pro protokol HTTP:

  `http://server:port/path?query`

2.1 Protokoly pro posílání dat

V této kapitole jsou vyjmenovány a popsány některé protokoly pro posílání dat (většinou souborů) přes síť Internet.

2.1.1 Protokol HTTP

HTTP (**Hypertext Transfer Protokol**) se používá pro distribuované informační systémy. Pro službu WWW se používá od roku 1990. První použitá verze HTTP 0.9 byla jednoduchá a zajišťovala pouze přenos dat po Internetu bez dalších doplňujících informací o těchto datech. HTTP verze 1.0 doplnila popisující informace do dotazů a odpovědí a byl k tomu použit formát MIME (Multipurpose Internet Mail Extension). Rozšířila tvar dotazu a odpovědi o standardizované doplňující informace o přenášených datech ve tvaru typ/podtyp. HTTP 1.0 je definováno v RFC 1945. S rozvojem služby WWW se objevovaly další požadavky na protokol HTTP. Především se jednalo o práci s hierarchickou strukturou proxy, využívání cache, požadavek na trvalé spojení mezi klientem a serverem a požadavky na virtuální servery. Tyto nedostatky řeší nová verze protokolu HTTP 1.1, která je definována v RFC 2068.

Metody protokolu

Metoda určuje druh služby, kterou klient od serveru požaduje. Metoda se uvádí velkými písmeny. Server nemusí vždy všechny metody podporovat. Při dotazu nepodporovanou metodou vrací chybové hlášení. Některé metody:

- **GET** - požadavek na poslání dokumentu určeného URL. V souvislosti s proxy se může metoda GET změnit na "podmíněný GET", který požaduje poslání dokumentu pouze za určitých podmínek, které jsou definovány v hlavičce dotazu.
- **POST** - metoda se používá v případě, kdy má cílový server přijmout data z požadavku. Skutečná funkce metody zavisí na URL s ní spojenou. Výsledkem metody POST může být poslání e-mailu, předání dat procesu, který data zpracuje, rozšíření databáze. Posílaná data nejsou nijak omezena a je možné tělo zprávy popsat v hlavičkách.
- **HEAD** - metoda je identická s metodou GET, server ale nemusí posílat tělo odpovědi. Metodu je možné použít k získání doplňkových informací o dokumentu. Často se používá k testování dostupnosti hypertextovych odkazů a času jejich poslední modifikace. Klient může získané hlavičky analyzovat a případně požádat o data novým dotazem GET. Například test, zda dokument není příliš dlouhý.
- **PUT** - požadavek na uložení posílaných dat pod specifikovaný URL na server. Takto uložená data budou dostupná např. naslednými dotazy GET. Uložení dat do souboru na server provádí přímo server.
- **DELETE** - požadavek na zrušení dokumentu na serveru. Rušený dokument je specifikován v URL.
- **TRACE** - metoda použitá k testování originálního serveru. Originální server má vratit klientovi kladnou odpověď bez dat.
- **OPTIONS** - dotaz na možnosti komunikace spojené s uvedeným URL. Metoda umožňuje klientovi určit možnosti a omezení spojené se zdrojem nebo schopnostmi serveru. Pokud je URL v dotazu ve tvaru "*", pak se jedná o dotaz na možnosti serveru.

HTTP servery používané v současné době podporují vždy metody GET, POST a HEAD.

Hlavičky HTTP

Hlavičky HTTP protokolu mají tvar podobný hlavičkám elektronické pošty:

```
název hlavičky: hodnota[;parametr=hodnota] CRLF
```

CRLF představuje znak pro nový řádek. Každá hlavička tedy začíná na samostatném řádku. Parametry jsou nepovinné, používají se jen u nekterých hlaviček.

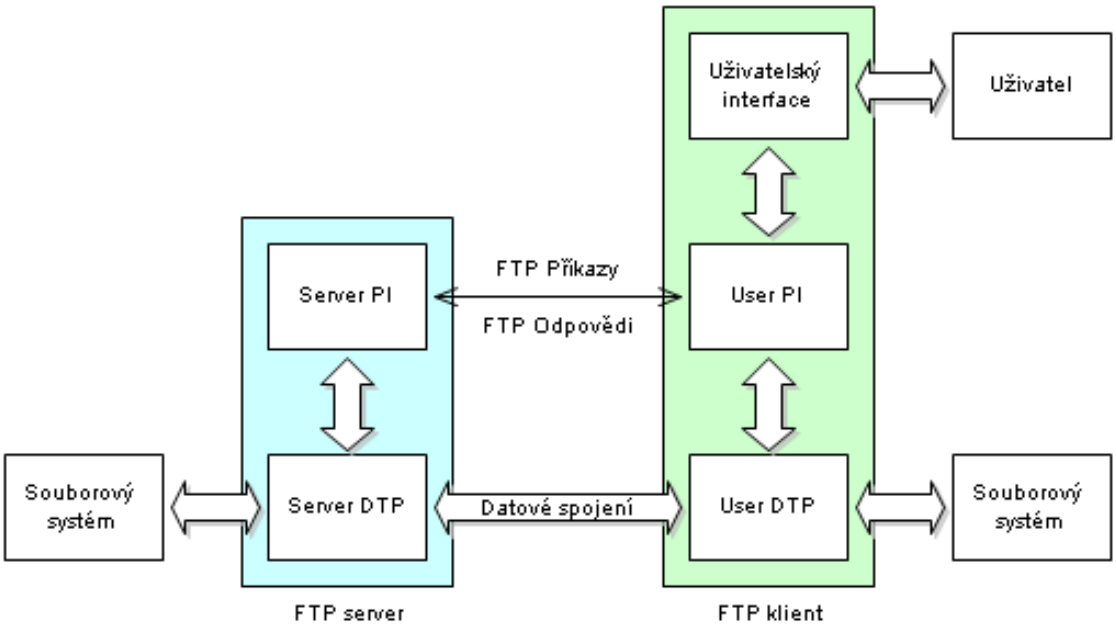
Hlavičky můžeme rozdělit do tří skupin:

- Obecné hlavičky poskytující univerzální informace o zprávě.
- Hlavičky dotazu nebo odpovědi, které popisují dotaz (odpověď).
- Hlavičky těla, které popisují tělo zprávy.

Na pořadí hlaviček nezáleží, ale ve standardu se doporučuje, aby hlavičky zprávy byly uspořádány podle svých tématických kategorií v uvedeném pořadí.

2.1.2 Protokol FTP

FTP (**File Transfer Protocol**) je protokol pro přenos souborů. Je specifikován v RFC 959. Pomocí FTP protokolu spolu komunikují FTP server a FTP klient. FTP klient se skládá z uživatelského rozhraní (**UI** - User Interface), uživatelského interpretu protokolu (**User PI** - Protocol Interpreter) a procesu, který přenáší data (**User DTP** - Data Transfer Process). Pomocí uživatelského rozharní komunikuje uživatel s klientem. Protocol Interpreter (User PI) vytváří řídící spojení se Serverem (Server PI), zasílá příkazy a ovládá proces, který přenáší data. Server naslouchá na portu a čeká na řídící spojení z User PI, přijímá příkazy, posílá odpovědi a ovládá Server DTP. Parametry datového spojení se specifikují pomocí příkazů. Server DTP vytváří datové spojení a poté přenáší a ukládá data. User DTP naslouchá na datovém portu a čeká na komunikaci se serverem. Server DTP může být také v pasivním režimu, ve kterém naslouchá na datovém portu a čeká na datové spojení, které vytváří User DTP. Datové spojení nemusí existovat po celou dobu, po kterou je klient k serveru připojen.



Obrázek 2.1: Model protokolu FTP

FTP Příkazy

Příkazy lze rozdělit do tří skupin:

1. Příkazy řízení přístupu (Access Control Commands)
2. Příkazy nastavující parametry přenosu (Transfer Parameter Commands)
3. Obsluhující příkazy (FTP Service Commands)

Příkazy řízení přístupu:

- **USER** - zadání uživatelského jména

- **PASS** - zadání uživatelského hesla
- **ACCT** - zadání uživatelského účtu (téměř se nepoužívá)
- **CWD** - změna aktuálního adresáře
- **CDUP** - změna aktuálního adresáře na nadřazený adresář
- **QUIT** - ukončení spojení

Příkazy nastavující parametry přenosu:

- **PORT** - specifikuje počítač a port pro datové spojení. Klient pošle tento příkaz a bude na daném portu čekat na datové spojení.
- **PASV** - žádá server o pasivní mód, tzn. že server bude poslouchat a klient bude iniciovat datové spojení.
- **TYPE** - určuje typ reprezentace dat, např. text nebo binární

Obsluhující příkazy:

- **RETR** - přenos souboru ze serveru
- **STORE** - přenos souboru na server
- **RNFR, RNTO** - přejmenování souboru
- **DELE** - smazání souboru
- **MKD** - vytvoření nového adresáře
- **RMD** - smazání adresáře (adresář musí být prázdný)
- **ABORT** - zrušení předchozího příkazu
- **PWD** - zjištění aktuálního pracovního adresáře
- **LIST** - získání seznamu souborů. K získání tohoto seznamu se musí otevřít datové spojení. Pokud je parametr tohoto příkazu adresář, získá se výpis tohoto adresáře. Pokud je to soubor, získají se informace o tomto souboru a pokud příkaz nemá parametr, je vrácen výpis aktuálního adresáře. Výpis příkazu list ale závisí na systému a je určen především pro člověka.
- **NLST** - podobný příkazu LIST, jen s tím rozdílem, že seznam vrací na každém řádku jen jméno souboru (adresáře) a žádné další informace. To je vhodnější pro zpracování počítačem.
- **SYST** - slouží k zjištění typu systému, na kterém běží FTP server.

Datová spojení a přenos souborů

Existují dva způsoby navázání datového spojení:

- **aktivní způsob navázání spojení** - spojení iniciuje server.
- **pasivní způsob navázání spojení** - spojení iniciuje klient.

Při aktivním způsobu iniciuje datové spojení server, při pasivním spojení je iniciátorem datového spojení klient. Pasivní způsob spojení se většinou používá, pokud je klient za firewallem. Při aktivním způsobu spojení, kdy je iniciátor server, požadavek na otevření datového spojení neprojde přes zmíněný firewall. V tomto případě se musí použít pasivní způsob, kdy datové spojení vytváří klient a tento požadavek přes firewall projde.

Při aktivním způsobu klient pošle příkaz PORT, kde jako parametr je IP adresa klienta a port, na kterém bude poslouchat. Server po obdržení tohoto příkazu vytvoří datové spojení právě podle parametrů příkazu PORT. Po vytvoření spojení klient zašle jeden z příkazů RETR, STORE nebo LIST a po datovém spojení se začnou posílat data.

Při pasivním způsobu klient zašle serveru příkaz PASV. Server na tento příkaz pošle odpověď, ve které bude IP adresa a port, na kterém bude server poslouchat. Klient vytvoří datové spojení na tuto IP adresu a port a pošle příkaz (např. RETR, STORE nebo LIST).

Příkazy PORT a PASV platí jen na následující datové spojení. Pokud je datové spojení ukončeno, musí se před otevřením nového datového spojení poslat nový příkaz PORT nebo PASV. Odpovědi od serveru

Na každý příkaz musí server poslat odpověď (reply). Tato odpověď se skládá z tříciferného čísla následovaného textem. Číslo je určeno pro zpracování počítačem, zatímco text je určen pro člověka. Text se totiž může na různých serverech lišit.

2.1.3 SFTP a SCP

Existují také protokoly pro šifrovaný přenos souborů, jako je SFTP a SCP. Tyto protokoly jsou využívány zejména tam, kde je zapotřebí bezpečného přenosu souborů. Posílaná data jsou šifrována a během přenosu jsou tedy zabezpečena proti někomu, kdo může odposlouchávat provoz na síti a takto neoprávněně získat přenášené soubory, nebo informace.

2.1.4 FSP

Je to protokol založený na UDP. Byl navržen jako náhrada Anonymních FTP serverů. Umožňuje uživatelům pouze na serverse přihlásit a přenášet vybrané soubory na svůj počítač. Oproti protokolu FTP má řadu výhod, ale již se téměř nepoužívá. Pro naše účely se nehodí. http://fsp.sourceforge.net/

2.1.5 Peer2peer sítě

Pro přenos souborů se dále používají protokoly, které jsou vystavěny na základě peer2peer sítí. To jsou sítě, ve kterých spolu komunikují uživatelé přímo. Je to jakýsi opak architektury klient-server, kde vždy na jedné straně stojí uživatel a na straně druhé centrální server, nebo servery. Uživatelé tak v architektuře klient-server spolu mohou komunikovat jen pomocí serveru. Protokoly peer2peer jsou pro nás méně využitelné, protože většinou slouží zejména ke sdílení souborů s více uživateli. Pokud budeme chtít poslat zásilku souborů jen jednomu člověku, který navíc ve chvíli, kdy zásilku chceme poslat není připojen k sítí, pak máme problém, protože v peer2peer síti není žádný server, na který bychom soubory mohli uložit.

2.2 Protokoly pro osobníkomunikaci uživatelů

V této části jsou popsány protokoly určené k osobní kolunikci mezi uživateli sítě Internet.

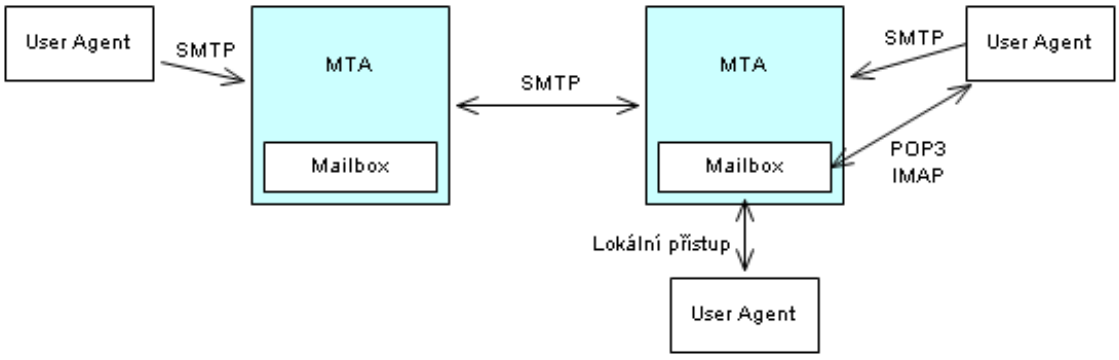
2.2.1 Elektronická pošta (e-mail)

Elektronická pošta je jednou z nejstarších aplikací na internetu. V dnešní době patří mezi základní služby. Původně byla navržena jen pro posílání jako jednoduchá služba služící k posílání krátkých textových zpráv.

Poštovní systémy se teoreticky skládají ze čtyř různých částí:

- **MUA** (**Mail User Agent**) - program, který vytvořenou zprávu předá k odeslání. Je to prostředí, ve kterám uživatel vytváří zprávy elektronické pošty.
- **MTA** (**Mail Transport Agent**) - program sloužící pro přenos zpráv elektronické pošty. Posílá zprávu dalšímu MTA.
- **MSP** (**Mail Submission Program**) - program, který odesílá zprávu, kterou mu předá MUA (Mail User Agent). Provádí kontrolu dat, korekci a předáni MTA.
- **MDA** (**Mail Delivery Agent**) - doručovací program, který přijaté zprávy třídí a ukládá do uživatelských schránek.

Dnes jsou úlohy, které v dřívější době tito “agenti” plnili většinou již integrovány do jednoho celku (programu). Například program **Sendmail** spojuje funkci MSP a MTA. **Sendmail** můžeme považovat za standard programů pro odesílání pošty. Také se používá **QMail**, **Postfix** nebo **Exim**. Jako MDA se používá **Procmail**, nebo **Maildrop**.



Obrázek 2.2: Architektura elektronické pošty

Formát e-mailu

Specifikace struktury elektronického dopisu je uvedena v specifikaci RFC 822 a specifikace MIME je definována ve specifikacích RFC 2045 až 2049. E-mailová zpráva má tyto části:

- **Obálka** - Při přenosu zprávy si MTA vytváří obálku, která definuje, kam bude zpráva doručena a komu má být vrácena pokud ji nelze doručit. Adresy na obálce se ve většině případů shodují s řádky From a To v hlavičce. Obálka je pro uživatele neviditelná.

- **Hlavičky** - končí prvním prázdným řádkem a obsahují parametry ve formátu:

  ```
  <klíč> : <parmetry>
  ```

  Standardní formát hlaviček je definován v RFC 2822. Jsou povoleny i nestandardní hlavičky, které začínají “X-”. Například X-Operating-System

  Hlavičky se dělí na **strukturované** a **nestrukturované**.

  Nestrukturované se počítačem nezpracovávají a jejich parametry mohou obsahovat libovolnou posloupnost znaků (např. Subject:).

  Strukturované jsou tvořeny tak, aby mohly být analyzované počítačem a jejich vytváření má určitá pravidla. Například zápis adres <adresa>, komentářů (komentář), textu obsahujícího speciální znaky ”text s mezerou”, doménového literálu [IP adresa]. Takto je například tvořena hlavička Received:

- Tělo zprávy - Řádky znaků podle US-ASCII tabulky, jejichž délka nesmí přesáhnout 998 znaků.

Protokol SMTP

SMTP (Simple Mail Transfer Protocol) je protokol, který předává e-mailové zprávy mezi počítači. Jeho úkolem je předávat e-maily spolehlivě a efektivně. Je definován v RFC 2821. Komunikace standardně probíhá na TCP portu 25 jako komunikace klient–server. Jako klient vystupuje MUA/MSP, který předává zprávu prvnímu MTA, nebo MTA, který předává zprávu dále. Klient naváže spojení na příslušný port a zadává SMTP příkazy. Server na ně odpovídá a sestavuje obálku zprávy podle které je zpráva dále posílána. Každá odpověď serveru začíná třímístným kódem výsledku operace.

Obrázek 2.3: Jak pracuje SMTP

Příkazy SMTP: V RFC 2821 je definováno, které příkazy musí akceptovat každý SMTP server. Jsou to tyto příkazy:

- EHLO <doména> - jako HELO, ale vypíše nestandardní akceptované příkazy a další parametry
- HELO <doména> - příkaz se používá k zahájení spojení
- MAIL FROM: <adresa> - přidání odesilatele do obálky
- RCPT TO: <adresa> - příkaz se používá k přidání příjemce do obálky. Více příjemců může být označeno vícenásobným použitím tohoto příkazu.

- DATA - příkaz zahajuje zasílání dat na server. Jako poslední data je odoslána tečka na samostatném řádku, která se vyhodnotí jako konec posílání dat.
- RSET - příkaz umožňuje předčasně ukončit aktuální transakci.
- NOOP - tento příkaz se používá na otestování spojení.
- QUIT - příkaz zavře komunikační kanál.
- VRFY <adresa> - příkaz požaduje od serveru potvrzení totožnosti užívatele.

příklad komunikace se SMTP serverem:

```
choch@thePipe:~$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 pocitac.domena.cz ESMTP Sendmail 8.13.6/8.13.4; Sun, 30 Apr 2006 20:08:13 +0200
HELO domena.cz
250 pocitac.domena.cz Hello localhost [127.0.0.1], pleased to meet you
MAIL FROM:<odesilatel@server.cz>
250 2.1.0 <odesilatel@server.cz>... Sender ok
RCPT TO:<adresat@server.cz>
250 2.1.5 <adresat@server.cz>... Recipient ok
DATA
354 Enter mail, end with ''.'' on a line by itself
hlavička e-mailu

tělo e-mailu
.
250 2.0.0 k3UI8DOj014298 Message accepted for delivery
QUIT
221 2.0.0 pocitac.domena.cz closing connection
Connection closed by foreign host.
```

Přijímání pošty

- Lokální - v dřívějších dobách se e-maily nejčastěji uchovávaly v lokální e-mailové schránce (Mailbox). Je to soubor, do kterého MDA (mail delivery agent) postupně ukládá přicházející zprávy. Tento soubor se většinou nachází v adresáři /var/spool/mail (na některých systémech je adresář jiný). Program na čtení a posílání e-mailů (MUA) tento soubor přečte a umožní zprávy zobrazovat a zpracovávat. To je ale možné jen když MUA běží na počítači, kde se tato e-mailová schránka nachází. To už ale dnes není tak časté.
- POP3 - dnes je nejčastěji používaný přístup k e-mailové schránce síťový protokol POP3 (Post Office Protocol verze 3), specifikovaný v normě RFC 1939. Je to jednoduchý protokol, kterým MUA přebírá zprávy z poštovního serveru. To znamená, že zpráva po doručení na serveru zůstává, dokud si ji odtud někdo nevyzvedne. Po převzetí zprávy ze serveru se zpráva na serveru může smazat, nebo ponechat. MUA potom tyto zprávy uchovává na lokálním počítači. Tento protokol je autentizovaný, ale nešifrovaný. Specifikace definuje metodu APOP, která používá k přenosu hesla od klienta k serveru md5 hash.

- IMAP (Internet Mail Access Protocol) je definován v normě RFC 3501. Stejně jako protokol POP3 byl navržen pro přístup ke zprávám uloženým v e-mailové schránce na poštovním serveru, ale je určen spíše pro dlouhodobější připojení. Zprávy zůstanou uloženy na serveru a průběžně se stahují dle potřeby. Rozdílem od POP je například podpora připojení více klientů současně, uchovávání stavu zprávy, možnost manipulace s více schránkami, stažení pouze hlavičky zprávy či prohledávání na straně serveru.

2.2.2 Instant messaging (IM)

Instant messaging je internetová služba, umožňující svým uživatelům udržovat seznam kontaktů a sledovat, kteří z jejich přátel jsou právě připojeni, a dle potřeby jim posílat zprávy a i jinak komunikovat. Většinou umožňují také posílání souborů. Nejsou však na tuto službu zaměřeni a často je toto posílání souborů nedokonalé. Instant Messanger je pak nástroj podporující nekteré z protokolů pro Instant Messaging.

ICQ

Jeden z prvních protokolů pro Instant messaging a také komunikační software, který tento protokol využívá. Jde o nejrozšířenější protokol, který má několik nevýhod, ale je velmi populární. Protokol je uzavřený, takže když někdo nepoužívá originální klientský program, při případné změně protokolu musí být jiné klientské programy upraveny.

Jabber

Jabber je otevřený komunikační protokol založený na XML. Jeho základ je standardizován. Hlavní výhodou je jeho otevřenost a nezávislost na jedné konkrétní firmě. Díky tomu existuje mnoho klientských programů pro nejrůznější operační systémy, dokonce i pro mobilní zařízení. Jabber také podporuje tzv. transporty, díky kterým umožňuje z jednoho klientského programu komunikovat i s uživateli jiných sítí (ICQ, AIM, Yahoo, MSN...). Transporty ale umožňují i jiné služby (předpověď počasí, televizní program...) K dalším přednostem patří i podpora šifrování. Klient je většinou jednoduchá aplikace a o většinu složitých funkcí se stará server.

2.2.3 IRC

Internet Relay Chat (IRC) je “chatovací” síť nebo protokol. IRC byla jednou z prvních možností komunikace v reálném čase po internetu. Uživatelé přes IRC komunikují převážně na tzv. kanálech. Uživatelé mohou být přítomni na více kanálech najednou a zároveň mezi sebou mohou komunikovat i individuálně.

Kapitola 3

Návrh aplikace

Aplikaci jsem rozdělil na dvě části a to na část, která se stará o přenos souborů z počítače *Computer 1* na *Server* a na část, která umožní uživateli přenést zásilku souborů ze *Serveru* na počítač *Computer 2*. *Server* musí být dostupný jak z počítače odesilatele, tak z počítače adresáta, protože jinak by přenos nebyl možný.

Snažil jsem se o to, aby mé řešení bylo použitelné pro co největší okruh uživatelů. Proto jsem volil komunikační protokoly, které jsou rozšířené a používané. Na Internetu existuje řada "poskytovatelů", kteří na svých serverech poskytují zdarma prostor určený na uložení webové stránky. Pro přenos a úpravu těchto webových prezentací se většinou používá protokol FTP. K přenosu zásilky na *Server* můžeme tedy využít tento protokol a pro přenos souborů na počítač *Computer 2* je možno použít protokol HTTP, který se používá k přístupu na webové stránky.

Zasílání upozornění o tom, že zásilka dorazila na server lze realizovat téměř jakýmkoli protokolem, který slouží ke komunikaci mezi uživateli počítačové sítě. Já jsem zvolil e-mail, protože je velmi rozšířený a téměř každý člověk připojený k Internetu e-mail používá. Aplikace ale může odesílat upozornění i více způsoby, než jen e-mailovou zprávou.

3.1 Odeslání souborů

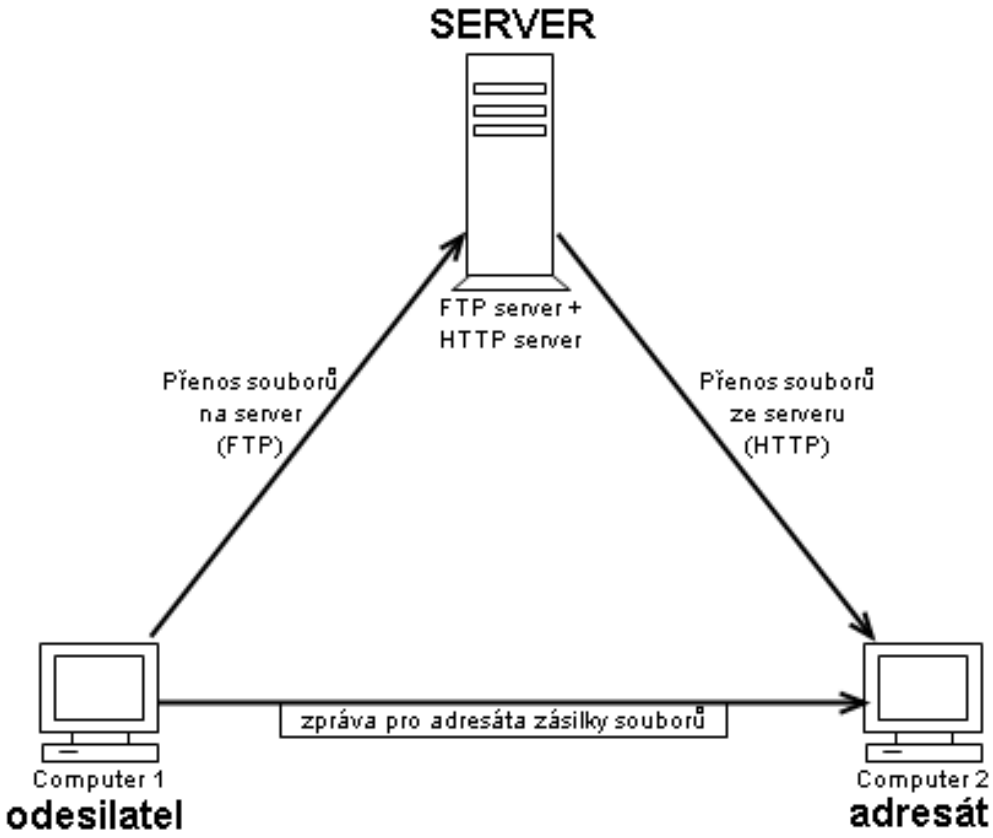
První část aplikace se stará o přenos souboru na server a o odeslání zprávy, která upozorní adresáta na to, že je zásilka k dispozici ke "stažení". Tato aplikace se nachází na počítači *Computer 1* a zasílá specifikovanou zásilku souborů na *Server*.

Aplikace od odesilatele musí získat tyto základní informace, aby mohla zásilku na server odeslat.

- na jaký FTP server se má zásilka odeslat - IP adresa, nebo doménové jméno serveru
- na jaký TCP port se připojit (standardní je 21)
- kde se nachází soubory, které jsou určeny k přenosu na server
- má se na server uložit komprimovaný soubor, nebo celý adresářový strom
- jaká metoda komprimace se má použít

Dále musí aplikace získat údaje potřebné k odeslání e-mailu, který adresátovi zásilky oznámí kde se jeho zásilka nachází a jak se k ní dostane.

- e-mailová adresa příjemce zprávy

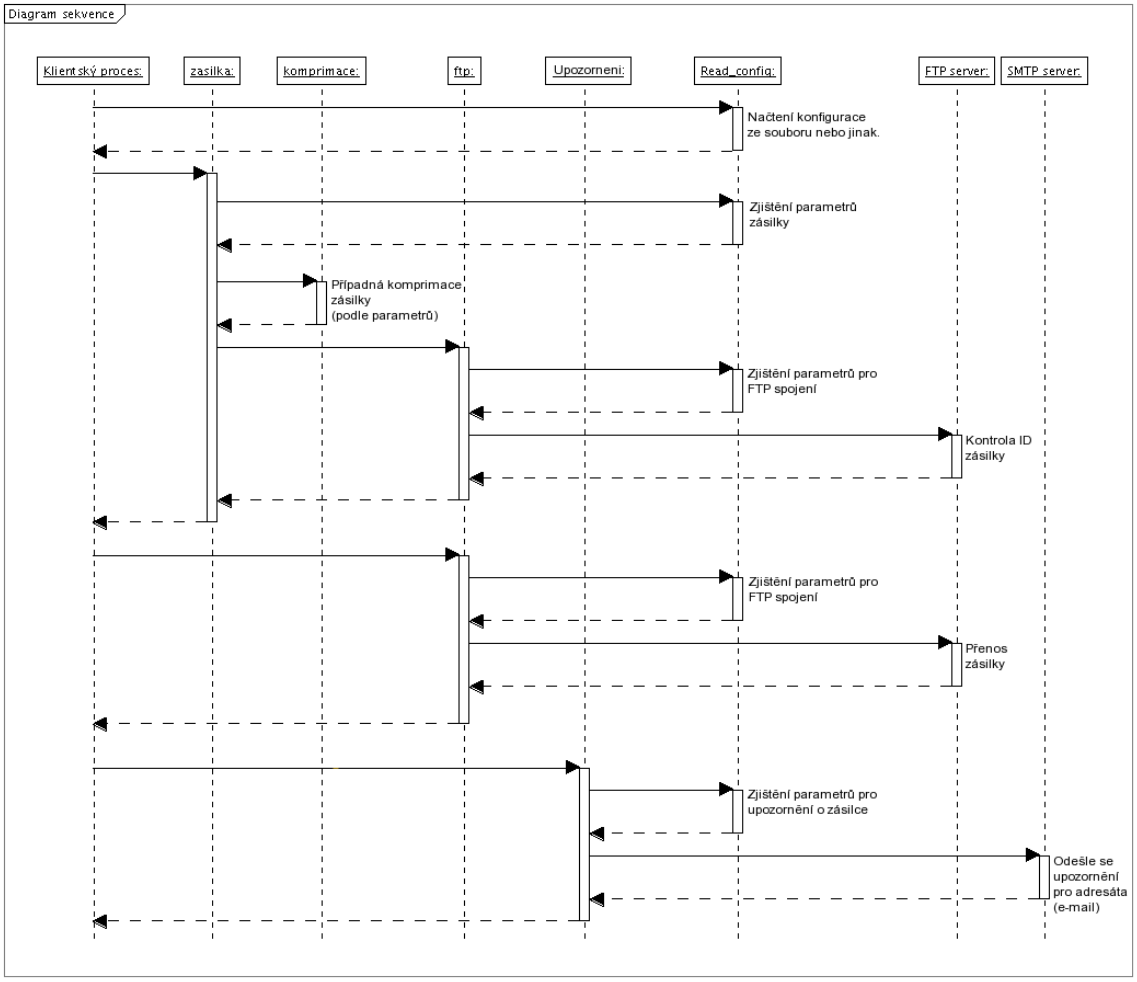


Obrázek 3.1: Schema činnosti aplikace

- další údaje zasílané v e-mailu jako Předmět (Subject), adresa odesilatele e-mailu a další údaje do zprávy včleněné.
- SMTP server, který může aplikace použít k odeslání e-mailu.
- URL na server, na který byla zásilka odeslána. Toto je adresa, na které si adresát může zásilku vyzvednout.

3.1.1 Postup při odesílání

Aplikace musí zjistit, zda má být odesílaná zásilka v komprimovaném tvaru. Pokud ano, tak provede komprimaci. Poté provede připojení k zadanému FTP serveru a pro zásilku vytvoří unikátní identifikátor, který ještě na serveru neexistuje. Pak musí být na serveru vytvořen adresář pojmenovaný jako identifikátor pro zásilku. Do tohoto adresáře se posléze nahraje buď komprimovaný soubor, vněmž je uložena celá odesílaná zásilka, nebo se musí vytvořit adresářová struktura přenášená z počítače Computer 1 a až bude vytvořena, mohou se do ní postupně přidat přenášené soubory. Dále je nutné odeslat upozornění pro příjemce zásilky, že zásilka dorazila na server. To aplikace provede tak, že pomocí zadaného SMTP serveru odešle e-mail, který bude obsahovat kód potřebný k přístupu do zásilky souborů, nebo přímo odkaz na adresu, na které si uživatel zásilku souborů bude moci vyzvednout.



Obrázek 3.2: Diagram sekvence pro odesílání zásilky

3.1.2 Tvorba náhodných identifikatorů

Náhodné identifikátory jsou potřeba k tomu, abychom od sebe mohli jednotlivé zásilky souborů na serveru odlišit a také proto, že na serveru budou soubory uložené v aresářích, které budou pojmenovány tak, že jejich jména budou obsahovat tento náhodný identifikátor. Identifikátory musí být vytvořeny tak, aby nedocházelo ke kolizím. V případě kolize by mohla být jedna zásilka souborů nahrána do už existující jiné zásilky a tak by byla porušena struktura obou. Identyfikátory se musí vytvářet způsobem takovým, že program nejprve vygeneruje náhodně první identifikátor a připojí se na server. Tam si ověří, že identifikátor ještě není použit pro jinou zásilku a až potom může tento identifikátor uznat za platný. Pokud však už zásilka s daným identifikátorem na serveru existuje, musí program znovu vytvořit náhodný identifikátor a ten znovu ověřit na serveru. Takto se identifikátor musí ověřit pokaždé. Proto by měl být řetězec používaný jako identifikátor dostatečně dlouhý, aby nedocházelo k častým kolizím. Čím delší řetězec je, tím méně často může ke kolizím docházet. délku řetězce volíme podle toho, kolik zásilek na serveru budeme uchovávat. Pro malé počty zásilek stačí kratší řetězce. Také je vhodné uvažovat, ze kterých znaků řetězce vytváříme. Čím větší počet znaků, tím více identifikátorů můžeme vytvořit, ale ne všechny znaky jsou použitelné.

3.1.3 Komprimace souborů (zabalení)

Komprmace dat je výhodná, protože komprimovaná data na serveru zaberou méně místa a můžeme pak do omezeného prostoru uložit více zásilek. Pro některé typy souborů může být koprimace velmi efektivní a ušetří čas při přenosu dat na server a ze serveru a také místo na serveru potřebné, ale pro jiný typ souboru může být komprimace méně účinná. Naříklad multimediální typy souborů, které už většinou jsou uložené v komprimované podobě jsou pro toto nevhodné. Existují také různé druhy komprese, různě náročné na procesorový čas. Výhodné je zvolit metodu komprese, která má dobré výsledky - prostor pro uložení dat je co nejmenší a zároveň komprimace netrvá delší dobu. V extrémním případě trvá komprimace velmi dlouho a výsledný soubor zabere více prostoru v paměti, než původní nekomprimovaná podoba.

3.1.4 Možnost ověření souborů pomocí md5 nebo sha hashe

Před odesláním souborů na server se vygenerují md5 nebo sha hashe, které mohou být odeslány spolu se zprávou pro adresáta. Toto opatření je vhodné pro zajištění nemožnosti změny souborů na serveru. Kdyby na serveru došlo ke změně souborů, adresát si může hashe znovu vygenerovat a zjistí, že se přenášené soubory změnily. Takto je vhodné ověřovat například data typu zdrojových textů nějakého programu, nebo balíky, vnichž jsou uloženy programy určené k instalaci. V těchto případech je totiž velmi důležité, aby bylo možno změny odhalit.

3.2 Druhá část aplikace (webové rozhraní)

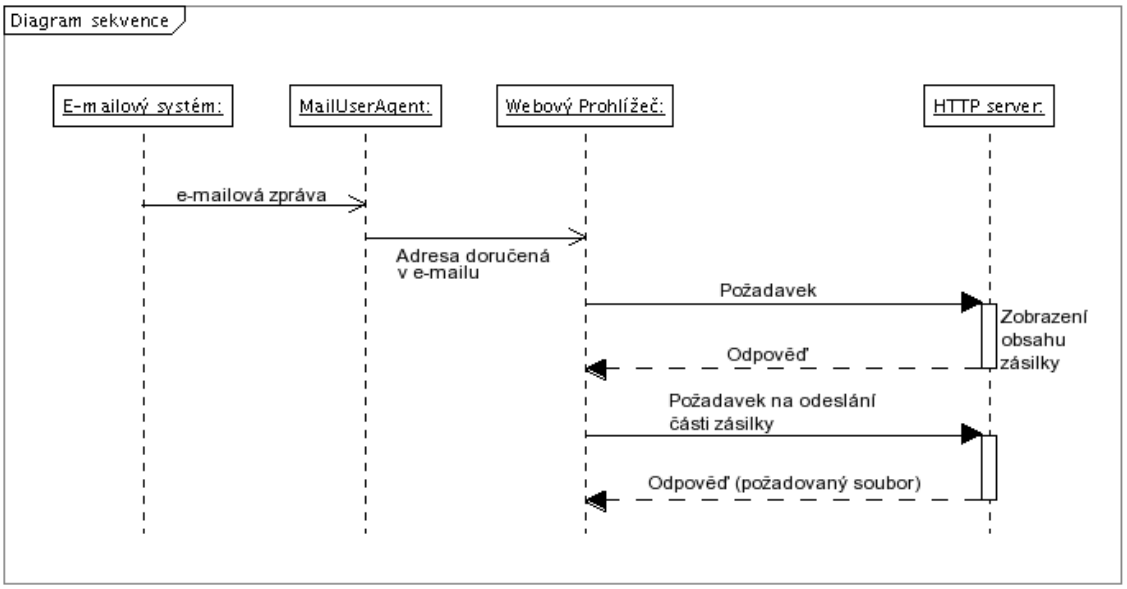
Tato část aplikace umožní adresátovi dostat se pomocí webového prohlížeče k zásilce souborů, která je pro něj na serveru připravena. Webové rozhraní umožní adresátovi po zadání kódu zásilky zobrazit soubory, případně adresářovou strukturu a umožní uživateli se v této struktuře pohybovat a také ukládat soubory na svůj počítač (*Computer 2*).

Adresátovi je doručen e-mail, který informuje o doručení zásilky na server. Zobrazí ho ve svém programu pro práci s e-mailovými zprávami a buď zjistí ze zprávy přístupový kód a adresu serveru, kde ho může zadat, aby se dostal k zásilce, nebo v zobrazeném e-mailu bude přímo odkaz na webovou stránku, kde se zásilka souborů zobrazí a adresát bude moci “procházet” adresářovou strukturou zásilky a ukládat jednotlivé soubory, nebo mu webové rozhraní umožní uložit celou zásilku, nebo její označenou část.

K tomu, aby si vyzvedl zásilku ze serveru potřebuje adresát adresu serveru, kde je zásilka uložena a přístupový kód k zásilce, nebo pouze adresu, která v sobě obsahuje přístupový kód k zásilce.

3.2.1 Platnost dat na serveru

Data na serveru budou uložena v adresářích, které mají jedinečné jméno. Platnost dat na serveru může být řešena tím, že vedle adresáře bude existovat soubor, jehož jméno bude obsahovat stejný identifikátor a bude uložen ve stejném adresáři, jako všechny adresáře se zásilkami. Tento soubor bude obsahovat informace o každé zásilce a bude v něm uložena platnost dat na serveru. V tomto souboru bude možno definovat Platnost dat pro celou zásilku a zvlášť pro každý soubor v zásilce uložený. Dále bude vytvořen skript, pomocí kterého bude možné kontrolovat platnost zásilek na serveru a provádět akce jako smazání zásilek, jejichž platnost vypršela.



Obrázek 3.3: Diagram sekvence pro přístup k zásilce

3.2.2 Ochrana proti stažení neautorizovanou osobou

Neautorizovaná osoba by neměla získat přístup k FTP serveru. To je zajištěno přihlašovacím jménem a heslem. Nevýhoda však je, že se toto heslo přenáší v nešifrované podobě a mohlo by být odhaleno neautorizovanou osobou Toto lze vyřešit použitím jiných protokolů, například SFTP. Vyzrazení hesla pro FTP přístup k serveru je kritičtější, než vyzrazení klíče k zásilce, protože přístup k FTP serveru umožní neautorizované osobě přístup ke všem uloženým zásilkám.

3.3 Výhody a nevýhody navrženého řešení

Zde shrnu přednosti a klady, ale také zápory mnou navrženého řešení.

3.3.1 výhody

- Uživatel se nemusí starat o detaily typu (jsem za firewalle. budu se moci ke svým souborům dostat?)
- Pokud příjemce dostane e-mail s adresou na zásilku souborů, nemusí si nic pamatovat a jen klikne na odkaz, nebo adresu zkopíruje do webového prohlížeče.
- HTTP server je ve většině případů dostupný, naproti tomu například posílání souborů přes některý z Instant Messangerů většinou nepůjde když jsem za firewallem, nebo za NAT (Network Adress Translation). To ynamená, že v podsíti máme jinou adresu.

3.3.2 nevýhody

- Kdo má přístup k serveru ať už fyzicky, nebo pomocí protokolu FTP, má přístup ke všem zásilkám na serveru uloženým.
- Pří přístupu na server pomocí protokolu FTP se přihlašovací jméno a heslo přenáší nešifrovaně a je možné je na síti odchytit pomocí specializovaných nástrojů.
- Data se vždy přenáší nešifrovaně.
- Při přístupu na server pomocí protokolu HTTP přenášíme klíč k přístupu do zásilky nešifrovaně.
- URL dresa pro přístup k zásilce souborů zůstane v historii navštívených stránek ve webovém prohlížeči. Pokud se pak ke stejnému webovému prohlížeči dostane někdo neoprávněně, může získat zásilku souborů z historie navštívených webových stránek.
- Spojení na FTP server nemusí vždy pracovat správně. například když jsme za firewallem.
- Program pro procházení a kontrolování a mazání zásilek, jejichž platnost vypršela bude velmi pomalý, protože musí otevřít soubor s informacemi o každé zásilce.

Kapitola 4

Implementace

Pro serverovou část - jsem použil programovací jazyk PHP, První část - *ufft.py* jsem implementoval v programovacím jazyku Python. Tento jazyk bývá někdy zařazován mezi takzvané skriptovací jazyky. Python je hybridní (víceparadigmatický) jazyk. To znamená, že umožňuje při psaní programů používat nejen objektově orientované paradigma, ale i procedurální a funkcionální, podle toho komu co vyhovuje. Kód programu je ve srovnání s jinými jazyky krátší. Je to program, který je využíván odesilatelem zásilky souborů.

Připojení k FTP serveru jsem implementoval za pomoci modulu ftplib, který implementuje většinu operací, které jsem pro přístup k FTP serveru protřeboval. Samozřejmě se našly i věci, které jsem musel vyřešit. Třída FTP například nemá metodu, pomocí které bych mohl jednoduše do seznamu uložit jména adresářů v nastaveném aktuálním adresáři. Toto jsem vyřešil tak, že jsem volal funkci, která vrací celý výpis - adresáře i soubory a z tohoto výpisu (seznamu řetězců) vyberu podle obsahu řetězce adresáře (jejich jména).

Dále jsem řešil problém, jak uložit adresářovou strukturu do zip archivu, když modul zipfile nepodporuje zadání cesty k adresáři, který má být rekurzivně do archivu přidán. Nejprve jsem toto chtěl řešit takovým, že bych nejprve do archivu vložil adresářovou strukturu a poté soubory, ale nenašel jsem žádnou metodu, pomocí které bych do archivu vložil prázdný adresář. Proto do archivu postupně vkládám soubory ze zásilky.

Při rozbalování zip archivu jsem narazil na menší problém. Vyhledem k tomu, že je nejprve musím vytvořit adresářovou strukturu, do které pak mohu ukládat jednotlivé soubory, musel první řešení, které mne napadlo bylo vytvořit celou adrsářovou strukturu a pak do ní vkládat jednotlivé soubory. To bych ale musel nejprve ze souboru načíst všechny položky (jména souborů), které tam byly uloženy a poté je načíst znovu, když bych je z archivu ukládal do adresářů. Poté mě napadlo jiné řešení. Pro každý soubor při vytváření adresáře, ve kterém je uložen nejprve musíme vytvořit adresář samotný. Program postupuje tak, že se nejprve pokusí vytvořit adresář, ve kterém je soubor přímo uložen. Pokud neuspěje, znamená to, že se musí pokusit vytvořit nadadresář, protože ještě neexistuje. Pokud uspějeme, vytvoříme i adresář u kterého jsme předtím neuspěli a pokud znovu neuspějeme znamená to, že musíme jít ještě výše v hierarchii adresářů, tedy blíže ke kořenovému adresáři do té doby, než, se vytvoření adresáře zdaří. Toto provádíme pro každý soubor, ale neznamená to nějaké významné zpomalení, protože pro většinu adresářů tato procedura uspěje.

Pro serverovou část jsem použil programovací jazyk PHP. Ta je určena ke zpřístupnění souborů na serveru uložených a je to jakési webové rozhraní pro příjemce zásilky. Zde jsem řešil problém jak zabezpečit soubory na serveru proti zneužití nevhodným obsahem zásilky. Například vložením HTML kódu, nebo dokonce PHP kódu jako souboru v zásilce. Stažení souboru jsem zajistil takto:

```
header(‘‘Content-Type:application/octet-stream’’);
```

```
header(‘‘Content-Disposition:attachment;filename=’’.$_GET[polozka]);
readfile(‘‘./’’.adresar.‘‘/’’.soubor);
```

Takto jsem zajistil také to, že se uživatel, který si soubor stahuje nedoví, kde přesmě na serveru je soubor uložen. To sice žádnou větší bezpečnost nezaručí, avšak odradí to některé uživatele, kteří rádi zkoušejí, jak by se dal systém poškodit.

Kapitola 5

Závěr

Cílem ročníkového projektu bylo navrhnout a vytvořit nástroj, který by usnadnil přenos souborů přes Internet. Navržené řešení spočívá ve vytvoření programu pro přeno souborů na server a ty pak jsou k dispozici na serveru ke stažení pomocí webového rozhraní. Nepodařilo se mi už z časových důvodů vytvořit grafické uživatelské rozhraní pro program používaný k odesílání dat na server. V ročníkovém projektu byly rozebrány možnosti přenosu souborů přes síť Internet pomocí dostupných nástrojů. Výsledkom praktické části je funkční program pro přenos zásilek souborů na FTP server a webové rozhraní pro přenos ze serveru pomocí protokolu HTTP.

Čerpal jsem z RFC [3], dále pak z internetové enciklopedie - Wikipedie [1], Stránek předmětu UNIX [2] a také Stránek Jabberu [5]. Dále jsem velmi často využíval webové stránky jazyka Python [4].

Literatura

[1] WWW stránky. Wikipedie - mezinárodní internetová encyklopedie. http://cs.wikipedia.org/.

[2] WWW stránky. Předmět UNIX na Fakulě informatiky MU. http://www.fi.muni.cz/~kas/p090/#archiv.

[3] WWW stránky. RFC dokumenty popisující Internetové protokoly. http://www.ietf.org/rfc.html.

[4] WWW stránky. Stránky programovacího jazyka Python. http://www.python.org.

[5] WWW stránky. Český portál sloužící k popularizaci a propagaci projektu Jabber. http://www.jabber.cz.