VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

KLIENT POP3 S PODPOROU TLS

POP3 CLIENT WITH TLS SUPPORT

PROJEKTOVÁ DOKUMENTACE
PROJECT DOCUMENTATION

AUTOR PRÁCE **TOMÁŠ KOHOUT**
AUTHOR

BRNO 2017

Obsah

1 Úvod **2**

2 Úvod do problematiky **3**
2.1 Elektronická pošta 3
2.2 Post Office Protol - verze 3 3
2.2.1 Autorizační stav 3
2.2.2 Transakční stav 4
2.2.3 Aktualizační stav 5
2.3 Rozšíření POP3 o TLS 5

3 Návrh **6**

4 Implementace **7**
4.1 Běh programu 7
4.2 Zpracování argumentů 7
4.3 Komunikace se serverem 7
4.4 Zajimavé pasáže implementace 8
4.4.1 Nové zprávy 8
4.4.2 Kombinace argumentů -n a -d 8
4.4.3 Mazání zpráv 8
4.4.4 Vytváření zabezpečeného socketu 8
4.4.5 Neblokující schránky 9

5 Návod k použití **10**

6 Závěr **12**

Literatura **13**

Kapitola 1

Úvod

Tato dokumentace pojednává o Post Office Protokol ve verzi 3. (dále jen POP3) a implementací klienta, který bude pomocí tohoto protokolu komunikovat s poštovním serverem. Mezi dalšími se tato dokumentace zabývá Transport Layer Security (dále jen TLS), což je kryptografický protokol pro šifrování zpráv zasílaných klientem a serverem. Tento protokol se implementuje jako rozšíření pro POP3.

Kapitola 2

Úvod do problematiky

2.1 Elektronická pošta

Elektronickou poštu používají každý den miliony lidí a denně se odešle 294 miliard emailů. [4] Svojí obrovskou popularitu získala tím, že je to bezplatný a jednoduchý způsob komunikace. Skrze elektronickou poštu se dnes zasílají obchodní nabídky, faktury nebo např. milostné dopisy.

Tyto zprávy můžeme číst skrze webový prohlížeč nebo pomocí různých poštovních klientů, kteří podporují jednak čtení zpráv a také odesílání nových zpráv. K tomuto účelu bylo vyvinuto množství komunikačních protokolů. V této dokumentaci se zaměříme pouze na protokol pro čtení zpráv a to Post Office Protocol ve verzi 3 (POP3).

2.2 Post Office Protol - verze 3

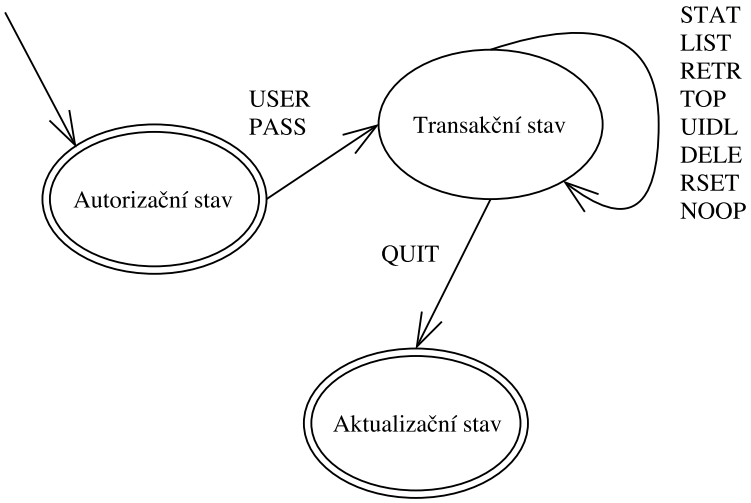
POP3 protokol využívá spolehlivý transportní protokol TCP a předvolený port 110. Díky POP3 jsme schopni získávat zprávy ze vzdáleného serveru, který podporuje komunikaci skrze POP3 a ukládat je do lokálního adresáře. Tyto zprávy můžeme také mazat ze vzdáleného serveru. Manipulace se zprávami probíhá tak, že se klient nejprve připojí k serveru, čímž vejde do tzv. autorizačního stavu, kdy je možné se pouze přihlásit. Po přihlášení se dostane do dalšího stavu nazvaného Transakční stav, kdy je možné získávat zprávy ze serveru nebo například mazat zprávy. Následeně se na žádost klienta dostane do dalšího a finálního stavu. Tento stav se nazývá Aktualizační stav a server až následně provádí změny provedené klientem, tedy vymaže zprávy označené pro smazání.

POP3 definuje sadu příkazů, které se používají pro komunikaci se serverem. V následujícím textu budou uvedeny a vysvětleny. Každý příkaz je ukončem znaky $\backslash r \backslash n$.

Server odesílá na tyto příkazy dva typy zpráv a to jednořádkové a víceřádkové odpovědi. Jednořádková odpověď je ukončena stejně jako příkaz od klienta, tedy $\backslash r \backslash n$. Víceřádková odpověď je ukončena $\backslash r \backslash n . \backslash r \backslash n$. Pokud řádek začíná tečkou a není to řádek ukončující víceřádkovou komunikaci, bude tato tečka zdvojena. [2] Klient se musí postarat o to, aby byla zdvojená tečka vymazána a byl tak dodržen Internet Message Standard [3].

2.2.1 Autorizační stav

Po úspěšném navázání spojení se nacházemí v autorizačním stavu. V tomto stavu jsou povoleny příkazy `USER, PASS, APOP`. Server odešle uvítací zprávu a čeká na autorizaci uživatele. Pokud se uživatel autorizuje, dostane se komunikace do dalšího stavu.



Obrázek 2.1: Stavový diagram

Pokud bylo připojení úspěšné, tak se na schránku, do které se klient přihlásil vloží zámek, aby se v daném čase mohl připojit vždy pouze jeden uživatel k jedné schránce. O tuto funkcionalitu se stará server.

Za příkazy `USER` a `PASS` následuje vždy mezera a uživatelské jméno, v případě příkazu `USER` nebo heslo v případě příkazu `PASS`.

2.2.2 Transakční stav

Pokud bylo přihlášení úspěšné, tak se komunikace dostává do nového stavu a to do transkačního stavu. V tomto stavu jsou povoleny následující příkazy `STAT, LIST, RETR, DELE, NOOP, RSET, UIDL, TOP`, kde `UIDL` a `TOP` jsou volitelné příkazy. Tento stav slouží pro manipulaci se schránkou, tedy ke stahování nových zpráv, mazání starých aj. Je zde uveden krátký výčet všech příkazů použitelných v transakčním stavu a také jejich způsob použití.

Příkaz `STAT` slouží ke zjištění počtu a velikosti všech zpráv uložených na straně server. Odpověď je jednořádková a obsahuje, v případě že nenastala žádná chyba `+OK` mezeru `počet zpráv` mezeru a `velikost všech zpráv` v bytech.

Příkaz `LIST` funguje podobně jako `STAT` s tím rozdílem, že odpověď je formou víceřádkové zprávy. První řádek obsahuje `+OK` a další řádky obsahují vždy číslo zprávy v mailboxu - od č. 1 až po maximální počet - a velikost zprávy v bytech.

Příkaz `RETR` se používá ke stažení zprávy ze serveru. Použití je následovné `RETR` mezera a číslo dané zprávy. Server na tento příkaz odpoví `+OK` mezera a velikost dané zprávy. Po prvním řádku následuje víceřádková zpráva obsahující celou zprávu v Internet Message Formatu [3].

Příkaz `DELE` slouží ke smazání zprávy z mailboxu. Odpověď od serveru je jednořádková. Příkaz se používá následovně `DELE` mezera a číslo zprávy, kterou chceme smazat.

Příkaz `NOOP` řekne serveru, aby dělal nic. Odpověď může být pouze `+OK`.

Příkaz `RSET` vrátí všechny změny provedené klientem do stavu jako po přihlášení. Odpověď serveru je jednořádková.

Příkaz `UIDL`, pokud je serverem podporovaný, vrátí víceřádkovou odpověď obsahující čísla zpráv a univerzální identifikační kód, který má každá zpráva jiný. Odpověď je víceřádková.

Přkaz `TOP`, pokud je serverem podporovaný, vrátí hlavičku zprávy a počet řádků v těle zprávy, pokud si to klient vyžádá. Použití je následovné. `TOP` mezera `číslo zprávy` mezera `počet požadovaných řádků z těla zprávy`. Odpověď je tedy víceřádková.

2.2.3 Aktualizační stav

Když se klient rozhodne ukončit práci se schránkou, tak pošle serveru příkaz `QUIT`, čímž se odhlásí a server provede případné smazání označených zpráv. Pokud server na ukončovací příkaz odpoví chybou, tak to znamená, že se označené zprávy neodstraní.

2.3 Rozšíření POP3 o TLS

POP3 protokol je zcela nechráněný proti případnému odposlouchávání. Proto vzniklo rozšíření tohoto protokolu, které přidalo nový port pouze pro šifrovanou komunikaci pomocí TLS a dále příkaz `STLS`, který řekne serveru v autorizačním stavu, aby začal komunikovat šifrovaně na standardním portu 110. [1]

Kapitola 3

Návrh

Cílem tohoto projektu je vytvořit klienta, který umožní správu emailové schránky, která je uložená na straně serveru. Pro komunikaci mezi aplikací klienta a serverem se bude používat Post Office Protocol verze 3 [2] (dále jen POP3). Dále je nutné implementovat rozšíření POP3s, které rozšiřuje POP3 o šifrovanou komunikaci. Klient musí podporovat příkaz `STLS`, který umožní navázat šifrované připojení, pokud to server umožňuje, na běžném portu pro POP3 komunikaci.

Klient musí umožňovat mazání zpráv ze serveru, stahování zpráv a stahování pouze nových zpráv, tzn. těch zpráv, které ještě nebyly staženy. Dálé musí umožnit použití vlastního certifikátu a to buď zadáním cesty k certifikátu v `.pem` formátu nebo cesty k adresáři obsahující certifikáty.

Kapitola 4

Implementace

Pro implementaci klienta byl použit jazyk `C++` ve verzi z roku 2011. Program je objektově orientovaný. Program nejprve zpracuje argumenty zadané uživatelem a následně vykoná jeho vůli.

4.1 Běh programu

Nedříve se vytvoří instance třídy `ParseParameters`, následně se pomocí metody `parse` zjistí parametry programu. Pokud je nastaven `-T`, vytvoří se instance třídy `TLSConnection` v opačném případě se vytvoří instance třídy `Connection`. Následně se v obou případech použije metoda `void authenticate()`, která se pokusí přihlásit k danému serveru. V případě úspěšného přihlášení se na základě parametrů použijí metody `int downloadMessages()` nebo `int deleteMessages()`. Tyto metody vrací počet stažených/smazaných zpráv. V případě korektního běhu programu se na standardní výstup vypíše počet stažených/smazaných zpráv. Pokud nastala v průběhu běhu chyba, tak se ukončí provádění programu a uživatel je informován inforamací na standardní chybový výstup.

4.2 Zpracování argumentů

O zpracování argumentů se stará třída `ParseParameters`. Stěžejní metodou této třídy je metoda `void parse(int argc, char *argv[])`. Jak musí být již zřejmé, tak na vstupu přijímá počet argumentů a pole řetězců, které obsahují argumenty zadané uživatelem. Toto pole se prochází v cyklu, kde se kontroluje, zda jsou zadány správné přepínače a korespondující hodnoty. V této třídě se kontroluje validita argumentů. V případě jakékoli nekonzistence dojde k vypsání nápovědy a ukončení programu.

4.3 Komunikace se serverem

Komunikaci se serverem obsluhují dvě třídy a to `Connection` a `TLSConnection`, kdy třída `Connection` zajišťuje nešifrované spojení a spojení navázané příkazem `STLS` a třída `TLSConnection` navazuje komunikaci se serverem již od počátku šifrovanou.

Obě tyto třídy rozšiřují abstraktní třídu `ConnectionInterface`, která poskytuje metody a virtuální metody potřebné pro komunikaci se serverem. Po implementaci metody `void establishConnection()` pro vytvoření připojení, `ssize_t readSock(char *buff,`

`int size)` pro čtení dat ze socketu a `void sendCommand(std::string cmd)` pro zasílání příkazu serveru je možné začít komunikovat se serverem.

Třída `Connection` vytvoří standardní BSD socket pro komunikaci se serverem a v případě přítomnosti argumentu -S přepne tento socket do šifrované podoby pomocí funkce `SSL_set_fd()`. Následně se pro komunikaci používá buď šifrovaná nebo nešifrovaná podoba komunikace.

`TLSConnection` se vytvoří v případě, že uživatel zadal argument -T pro vytvoření šifrované komunikace již od počátku běhu komunikace se serverem.

4.4 Zajimavé pasáže implementace

Zprávy se ukládají do složky definované programovým argumentem `-o`. Jednotlivé soubory jsou pojmenované čísly v desitkové soustavě. Nejnovější zpráva má nejvyšší číslo a naopak nejstarší zpráva má číslo nejnižší.

4.4.1 Nové zprávy

Pro zjištění novosti zpráv je použita hashovací funkce, která pro každou zprávu vytvoří odpovídající hash. Tyto informace jsou ukládány do souboru `file` v adresáři `.cache`, který se nachází v adresáři pro ukládání zpráv definovaný uživatelem.

Nevýhodou této implementace je především časová náročnost. Výhodou je možnost používat parametr `-n` i v případě, kdy server nepodporuje příkaz `UIDL`. Při běžném stahování zpráv není patrné zpoždění.

4.4.2 Kombinace argumentů -n a -d

Při použití argumentů `-n` a `-d` současně, dojde ke stažení nových zpráv a následně k jejich smazání se serveru.

4.4.3 Mazání zpráv

Argument `-d` maže pouze ty zprávy, které již byly jednou staženy. Po použití argumentu `-d` dojde ke smazání obsahu souboru `file` v adresáři `.cache`.

4.4.4 Vytváření zabezpečeného socketu

Nejprve se vytvoří normální schránka pomocí standardních funkcí pro vytvoření schránky. Jakmile máme tuto schránku vytvořenou, tak inicializujeme chybové výpisy SSL, inicializujeme knihovny SSL a načteme šifrovací algoritmy obsažené v OpenSSL (`SSL_load_error_strings()`, `SSL_library_init()`, `OpenSSL_add_all_algorithms()`). Poté vytvoříme kontext připojení pomocí funkce `SSL_CTX_new()`. Následně načteme certifikáty a to buď defaultní nebo uživatelem zadané. Poté pomocí funkce `SSL_new` vytvoříme SSL vrstvu, které pomocí funkce `SSL_set_fd()` nastavíme dříve vytvořenou schránku. Poté pomocí funkce `SSL_connect()` navážeme zabezpečené spojení. Následně pomocí funkce `SSL_get_peer_certificate()` získáme certifikát od serveru, protože funkce `SSL_get_verify_result()` nezjistí chybu, pokud server certikát neposkytne. Pokud je získaný certifikát prázdný, tak se připojení ukončí, jinak se zavolá funkce `SSL_get_verify_result()`, která zkontroluje správnost předloženého certifikátu.

4.4.5 Neblokující schránky

Klient pracuje s neblokujícími schránkami, takže se nemůže stát, že při nastavení špatného portu dojde k zaseknutí na funkci `connect()` nebo při čekání na odpověď, která nikdy nepřijde. Při připojování na server je nastaven 10 sekundový limit. Pro čekání na data je nastavený 5 sekundový limit.

Při použití šifrovaného spojení je potřeba přepnout schránky do neblokujícího režimu až po vytvoření šifrovaného spojení.

Kapitola 5

Návod k použití

Pro přeložení programu je potřeba mít nainstalovanou `Openssl` knihovnu a to od verze 1.0.2 po nejnovější. Dále je třeba mít překladač `g++`, který podporuje jazyk `C++` ve verzi z roku 2011.

Pokud splňujete výše uvedené, otevřete terminál ve složce s programem a spusťte příkazem `make` překlad. Po přeložení programu spusťte program jak je popsáno níže.

Tento program je sestavený z následujících zdrojových souborů:

`main.cpp, Connection.cpp/h, ConnectionInterface.cpp/h, Error.cpp/h, FileManipulator.cpp/h, ParseParameters.cpp/h, TLSConnection.cpp/h`.

popcl <server> -a <auth_file> -o <out_dir> [-p <port>] [-d] [-n] [-T|-S [-c <certfile>] [-C <certaddr>]] [-h|–help]

Povinné argumenty

<server>
IPv4 nebo IPv6 adresa, nebo doménové jméno

–a <auth_file>
soubor s autentifikačními údaji

–o <out_dir>
adresář pro ukládání zpráv

Volitelné parametry

–p <port>
zmení výchozí nastavení portu

–n
stáhne pouze nové zprávy, tedy ty které ještě nebyly staženy

–d
smaže již stažené zprávy

–n -d
stáhne nové zprávy a následně smaže již stažené zprávy

–T
zapne šiforvání celé komunikace

–S
naváže nešifrované spojení a pomocí příkazu STLS přejde na šifrované spojení

–c <certfile>

cesta k souboru, který obsahuje certifikát s koncovkou .pem

–C <certaddr>

cesta k adresáři obsahující adresář certikátů

–h|–help

vypíše nápovědu a skončí

Parametry -c a -C je možné používat společně nebo každý zvlášť, ale pouze v případě, kdy je použitý parametr `-S` nebo `-T`.

Kapitola 6

Závěr

Cílem projektu bylo vytvořit síťovou aplikaci a získat znalosti o vytváření síťové aplikace podle definovaného protokolu. Bylo nezbytné získat řadu znalostí a následně je aplikovat v praxi. Nejvíce času mi zabralo samotné programování daného klienta. Pochopil jsem jak funguje POP3 protokol a jak se pracuje s nezabezpečenými i zabezpečenými schránkami. Ke zdárné immplementaci mi pomohly předešlé zkušenosti z předmětu IPK a IJA.

Literatura

[1] C. Newman: Using TLS with IMAP, POP3 and ACAP. RFC 2595, RFC Editor, june 1999.
URL https://tools.ietf.org/html/rfc2595

[2] Myers, J. G.; Rose, M. T.: Post Office Protocol - Version 3 (POP3). RFC 1939, RFC Editor, may 1996.
URL https://tools.ietf.org/html/rfc1939

[3] Peter W. Resnick: Internet Message Format. RFC 5322, RFC Editor, october 2008.
URL https://tools.ietf.org/html/rfc5322

[4] Pingdom: Internet 2010 in numbers. [Online; navštíveno 14.11.2017].
URL http://royal.pingdom.com/2011/01/12/internet-2010-in-numbers/