VYSOKÉ UČENÍ TECHNICKÉ V BRNE
FAKULTA INFORMAČNÝCH TECHNOLOGIÍ

SÍŤOVÉ APLIKACE A SPRÁVA SÍTI 2021/2022

KLIENT POP3 S PODPOROU TLS

IMPLEMENTAČNÁ DOKUMENTÁCIA

Marek Miček (xmicek08) 13.10.2021

Obsah

1 Úvod

Cieľom tohto projektu bolo v jazyku C/C++ vytvoriť klientskú aplikáciu umožňujúcu komunikáciu so serverom pomocou POP3 protokolu s možným prechodom na TLS komunikáciu. Klient užívateľovi umožňuje sťahovať prípadne mazať e-mailové správy zo serveru. Táto dokumentácia popisuje postup pri implementácii tohto klienta.

2 Uvedenie do problematiky

2.1 Klient-Server komunikácia

Pri architektúre typu Klient-Server sa používa systém, ktorý sa skladá zo SW klienta a SW servera. Klient predstavuje iniciátora tejto komunikácie, pretože zasiela svoje požiadavky na server a čaká na jeho odpoveď. Zvyčajne sa pripája na malé množstvo serverov v jeden čas a typicky sa ovplyvňuje koncovými užívateľmi, ktorý používajú grafické užívateľské prostredie. Na druhej strane server je pasívny prvok tejto komunikácie a čaká na požiadavky od klienta. Tieto požiadavky prijíma, spracováva a zasiela odpovede klientovi. Väčšinou akceptuje pripojenia od väčšieho množstva klientov a typicky sa neovplyvňuje koncovými užívateľmi (viz [1]).

2.2 POP3 protokol

Komunikácia medzi klientom a serverom sa musí riadiť určitými pravidlami, ktoré definuje príslušný protokol. V našom prípade sa budeme riadiť POP3 protokolom. Každá textová požiadavka klienta sa musí sa skladať z príslušného príkazu, tela príkazu a ukončovacími znakmi "\r\n" (napr. "USER marek\r\n"). Medzi základné príkazy tohto protokolu patria USER, PASS, LIST, RETR, DELE, STAT, QUIT. Ich sémantika je nasledovná:

USER <username>\r\n	posiela meno užívateľa
PASS <password>\r\n	posiela heslo užívateľa
LIST\r\n	vráti zoznam správ na serveri s ich ID a veľkosťou
RETR <ID>\r\n	vráti správu s príslušným ID
DELE <ID>\r\n	vymaže správu s príslušným ID
STAT\r\n	vráti celkový počet správ a celková veľkosť
QUIT\r\n	koniec komunikácie so serverom

Server taktiež odpovedá textovými správami, pričom v prípade úspechu začína reťazcom “+OK”, v opačnom prípade “-ERR”. Odpovede sú buď jednoriadkové alebo viacriadkové. Jednoriadkové končia znakmi \r\n, viacriadkové znakmi \r\n.\r\n. Celá komunikácia medzi klientom a serverom je stavová a skladá sa z nasledujúcich stavov. Po tom ako sa naviazalo TCP spojenie a server poslal úvodný pozdrav informujúci užívateľa, že je pripravený komunikovať, prechádza sa do stavu autorizácie. Tu sa klient musí identifikovať pomocou svoji prihlasovacích údajov a príkazmi USER a PASS. V prípade oprávneného prístupu server získa všetky prostriedky potrebné k prístupu do schránky užívateľa a prejde sa do transakčného stavu. V tomto stave klient posiela svoje požiadavky na manipuláciu so svojimi správami (sťahovanie, mazanie a podobne). Ak klient pošle príkaz QUIT, tak server prejde do stavu UPDATE, kde uvoľní všetky prostriedky potrebné k manipulácii s klientovou schránkou rozlúči sa s užívateľom (viz [2]).

3 Návrh a implementácia programu

Program bol implementovaný v jazyku C++, pričom sa využil objektový návrh, kvôli lepšej udržateľnosti a znovupoužiteľnosti. Takmer všetky potrebné funkcie a atribúty sú zahrnuté v triede POP3_Client, pričom celý program je riadený z funkcie `main`.

3.1 Spracovanie argumentov

Na začiatku sa spracujú argumenty zadané používateľom. Toto prebieha vo funkcii `ParseArgs`, kde sa pomocou cyklu `for` spracuje každý argument a získané poznatky sa uložia do štruktúry `Popcl_args` na ďalšie použite. V prípade chybne zadaného argumentu alebo zadaného argumentu -h/--help bude užívateľovi vypísaná nápoveda na použite programu. Avšak argument -h/--help sa nesmie kombinovať so žiadnym iným argumentom. V tejto funkcii sa tiež kontroluje prítomnosť súboru s autentifikačnými údajmi a adresáru do ktorého sa budú stiahnuté správy ukladať. V tejto funkcii sa tiež kontroluje, či užívateľ nezadal parameter -n a -d súčasne. Tento klient pomocou týchto parametrov buď sťahuje nové správy, alebo vymaže všetky správy zo servera, no nesmie ich zadať súčasne. Sémantický význam jednotlivých argumentov je popísaný v kapitole č.4.

3.2 Pripojenie na server

Pred samotným pripojením sa musí určiť číslo portu na serveri. V prípade, že užívateľ nešpecifikoval žiadny port, tak tento port sa špecifikuje nasledovne. Pokiaľ si užívateľ nevyžiadal šifrovanú komunikaciu (nezadal argument -T) tak sa bude pracovať s portom 110. V opačnom prípade s portom 995 (viz [3]). Následne sa v závislosti na argumentoch -S a -T rozhodne akým spôsobom sa klient pripojí na server. Najjednoduchšia je samozrejme nešifrovaná varianta komunikácie so serverom, kedy sa volá funkcia `Connect_Server`. Na pripojenie sa využíva objekt BIO z knižnice <openssl/bio.h>. Táto knižnica poskytuje rozhranie pre rôzne typy komunikácie a zabezpečenia. V mnohých prípadoch je jej použitie jednoduchšie ako pri knižnici BSD sockets (viz [4]). V prípade prítomnosti argumentu -T sa volá funkcia `Connect_Server_Sec`. Keďže sa jedná o šifrovanú komunikáciu, je potrebné načítať úložisko s potrebnými certifikátmi pre overenie certifikátov zaslaných serverom, keďže bez toho k šifrovanej komunikácii nemôže dôjsť. Toto zaisťuje funkcia `Load_Certificates`. Podobný postup sa uplatní aj v prípade, že užívateľ zadá argument -S, akurát táto komunikácia prebieha na porte 110 a na začiatok sa naviaže nešifrované spojenie cez funkciu `Connect_Server` a až potom sa príkazom STLS prejde na šifrovanú variantu (viz [5]).

V oboch prípadoch šifrovanej komunikácie sa kontroluje, či adresa servera nie je typu IPv6. Pokiaľ je, tak sa na pripojenie k serveru použije knižnica BSD socket.

3.3 Autentifikácia užívateľa

Po pripojení na server nasleduje autentifikácia pomocou funkcie `Login_Server`. Prihlasovacie údaje sa načítajú zo súboru daného argumentom -a, pričom sa postupne pošlú príkazy USER a PASS s užívateľovým menom a heslom.

3.4 Zasielanie príkazov na server

Jednotlivé príkazy sa posielajú pomocou funkcie `Send_Server_Command`. Tu sa kontroluje počet bytov zaslaných na server. Hodnota rovná alebo menšia nule signalizuje chybu a v takom prípade sa program pokúsi o opätovný zápis pomocou vstavanej funkcie BIO_should_retry (viz [6]. Ak sa aj v tomto prípade vráti záporná hodnota tak sa signalizuje chyba.

3.5 Obdržanie odpovede zo serveru

Získanie odpovede zo serveru prebieha vo funkcii `Get_Server_Response` v nekonečnej `while` smyčke, ktorá sa ukončí až v prípade obdržania celej odpovede zo serveru. Koniec odpovede sa detekuje dvoma spôsobmi. Ak sa jedná o viacriadkovú odpoveď (v tejto aplikácii len odpoveď na príkaz RETR) klient sa snaží detekovať reťazec “\r\n.\r\n”. O ostatných príkazoch sa očakáva jednoriadková odpoveď ukončená “\r\n”. Celá odpoveď je uložená v triednom atribúte `Server_Response`.

3.6 Sťahovanie správ

Táto funkcionalita je zabezpečená funkciou `Download_Messages`. Najskôr sa pomocou funkcie `Get_STAT` zistí celkový počet správ na serveri a potom pomocou `for` cyklu sa posiela na server príkaz RETR presne toľkokrát, koľko tam je správ. V tomto cykle sa pre každú správu vytvorí nový textový súbor pomenovaný nasledovne: OutDir + “/” + User + “_message_” + ID + “.txt”.
OutDir je adresár daný argumentom -o, User je meno používateľa a ID je číslo správy. Predtým ako sa obsah správy zapíše do súboru, klient odstráni prvý riadok, ktorý obsahuje pozitívnu odpoveď serveru a posledný riadok s ukončovacím reťazcom “\r\n.\r\n”.

3.7 Sťahovanie nových správ

Po každom dokončení sťahovania správ zo serveru sa uloží do súboru `ID_Database_Fiel.txt`, vytvorenom v adresári, kde je náš program, jedinečné UID správy, ktoré sa stiahli ako posledné (správa mala najvyššie MessageID pri klasickom sťahovaní). Následne sa vo funkcii `Download_New_Messages` pomocou `for` cyklu prechádzajú správy od najvyššieho MessageId po najnižšie a pre každú správu posiela príkaz UIDL MessageID, ktorý vráti UIDL aktuálnej správy. Toto UIDL sa porovná s UIDL zo súboru a pokiaľ sa rovnajú, tak sa pracuje so správou, ktorá už je stiahnutá a `for` cyklus sa ukončí. No problémom je, že nie každý POP3 server príkaz UIDL podporuje. Ak ho nepodporuje tak sa uloží do vyššie spomenutého súboru najvyššie MessageID. No v takomto prípade môže vznikať problém, ktorý je popísaný v sekcii Obmedzenia. Rovnako ako pri obyčajnom sťahovaní sa pre každú správu vytvorí nový textový súbor pomenovaný : OutDir + "/" + User + "_new_message_" + counter + ".txt". Význam položiek je rovnaký ako u súborov pre klasické sťahovanie, iba premenná counter uchováva počítadlo nových správ. Taktiež sa odstraňuje prvý a posledný riadok správy. Na záver sa aktualizuje hodnota UID/ID v súbore `ID_Database_Fiel.txt`.

3.8 Vymazávanie správ

Zabezpečuje funkcia `Del_Command`. Najskôr sa volaním funkcie `Get_STAT` zistí počet všetkých správ na serveri a vo `for` cykle sa posiela príkaz `DELE` presne toľko krát, koľko je na serveri správ.

3.9 Ukončenie spojenia

Implementované vo funkcii `Quit_Command`, kedy sa jednoducho zašle na server príkaz QUIT a ukončí sa spojenie so serverom.

4 Spustenie programu

Program sa spúšťa nasledovne: ./popcl <server> [-p <port>] [-T|-S [-c <certfile>] [-C <certaddr>]] [-d] [-n] -a <auth_file> -o <out_dir>

- Povinne je uvedený názov <*server*> (IP adresa, nebo doménové meno) požadovaného zdroja
- Voliteľný parameter -*p* špecifikuje číslo portu <*port*> na serveru
- Parameter -*T* zapína šifrovanie celej komunikácie (pop3s), ak parameter nie je uvedený použije sa nešifrovaná varianta protokolu
- Parameter -S naviaže nešifrované spojení so serverom a pomocou príkazu STLS (RFC 2595) prejde na šifrovanou variantu protokolu
- Voliteľný parameter -*c* definuje súbor <*certfile*> s certifikáty, ktorý sa použije pre overenie platnosti certifikátu SSL/TLS predloženého serverom (len s parametrom -*T*, alebo -*S*)
- Voliteľný parameter -*C* určuje adresár <*certaddr*>, v ktorom sa majú vyhľadať certifikáty, ktoré sa použijú pro overenie platnosti certifikátu SSL/TLS predloženého serverom. (len s parametrom -*T*, alebo -*S*.)
- Ak nie je uvedený parameter -*c* ani -*C*, tak sa použije úložisko certifikátov získané funkciou *SSL_CTX_set_default_verify_paths()*
- Pri použití parametru -*d* sa zašle serveru príkaz pro zmazanie správ, klient vyžaduje, aby sa nekombinoval s parametrom -n, vymazané správy sa predom neuložia do lokálneho adresára, takže je potrebné používať daný parameter opatrne
- Pri použití parametru -*n* sa bude pracovať len s novými správami, klient vyžaduje, aby sa nekombinoval s parametrom -d
- Povinný parameter -*a* <*auth_file*> udáva súbor s autentifikačnými údajmi
- Povinný parameter -*o* <*out_dir*> špecifikuje výstupní adresár <*out_dir*>, do ktorého má program stiahnuté správy uložiť

4.1 Ukážka použitia programu

- Stiahnutie všetkých správ zo serveru v nešifrovanej komunikácii

./popcl pop3.gmail.com -a AuthFile -o OutDir
Number of messages which were downloaded: 3

- Stiahnutie nových správ zo serveru v nešifrovanej komunikácii

./popcl pop3.gmail.com -a AuthFile -o OutDir -n
Number of new messages which were downloaded: 1

- Sťahovanie nových správ cez šifrované spojenie TLS

./popcl pop3.gmail.com -a AuthFile -o OutDir -n -T
Number of new messages which were downloaded: 2

- Sťahovanie správ s príkazom STLS a zadaným adresárom s vlastnými certifikátmi

./popcl 78.125.204.109 -a AuthFile -o OutDir -n -S -C /certs
Number of new messages which were downloaded: 1

- Vymazanie všetkých správ zo serveru (Pozor !!! Správy sa pred vymazaním nesťahujú)

./popcl pop3.gmail.com -a AuthFile -o OutDir -d

5 Obmedzenia programu

V prípade, že POP3 server nepodporuje príkaz UIDL, tak ako bolo spomenuté v kapitole 3.7, ukladá sa do vytvoreného súboru `ID_Database_Fiel.txt` MessageID s najvyššou hodnotou (keďže pri sťahovaní nových správ sa prechádzajú správy na serveri odzadu). No v takomto prípade môže nastať problém. Ak sa totiž zo serveru odstráni práve posledná správa s najvyšším MessageID a potom sa pošle napríklad nová, tak práve táto nová bude mať MessageID, ktoré je uložené v klientskom súbore, takže sa bude na túto správu pozerať ako na starú a pritom tomu tak nie je. Preto je vždy lepšie pracovať s UIDL, lebo je vždy originálne. No v zadaní tohto projektu je napísané, že sa zo serveru za použitia argumentu -d vymažú všetky správy, takže sa teoreticky nemôže stať situácia s MessageID spomenutá vyššie. No treba na to myslieť z globálneho hľadiska, pretože programy by mali byť čo najviac použiteľné a prenositeľné.

Referencie

[1] Klient-server – Wikipédia. [online]. [cit. 13.10.2021]
Dostupné z: https://sk.wikipedia.org/wiki/Klient-server

[2] Myers, J.; Rose, M.: Post Office Protocol - Version 3 [Internet Requests for Comments]. [b.m.]: RFC Editor, May 1996 [cit. 13.10.2021] RFC, 1939
Dostupné z: https://datatracker.ietf.org/doc/html/rfc1939

[3] Mail Ports for POP3, IMAP and SMTP. [online]. [cit. 13.10.2021]
Dostupné z: https://billing.precedence.com.au/billing/knowledgebase/70/Mail-Ports-for-POP3-IMAP-and-SMTP.html

[4] bio. [online]. [cit. 13.10.2021]
Dostupné z: https://www.openssl.org/docs/man1.1.0/man3/bio.html

[5] Ballard, K.: Secure programming with the OpenSSl API [online]. 22.7.2004 Posledná zmena 16.8.2018 [cit. 13.10.2021]
Dostupné z: https://developer.ibm.com/tutorials/l-openssl/