Simpler POP3 Client mit TLS Support (29)

David Fischer (03), 5CHIF

April 2021

Contents

1 Einleitung

Laut Angabe war das Ziel dieser Aufgabe, einen POP3 Client inklusive Unterstützung für GnuTLS [3] zu erstellen.

2 Klassendiagramm

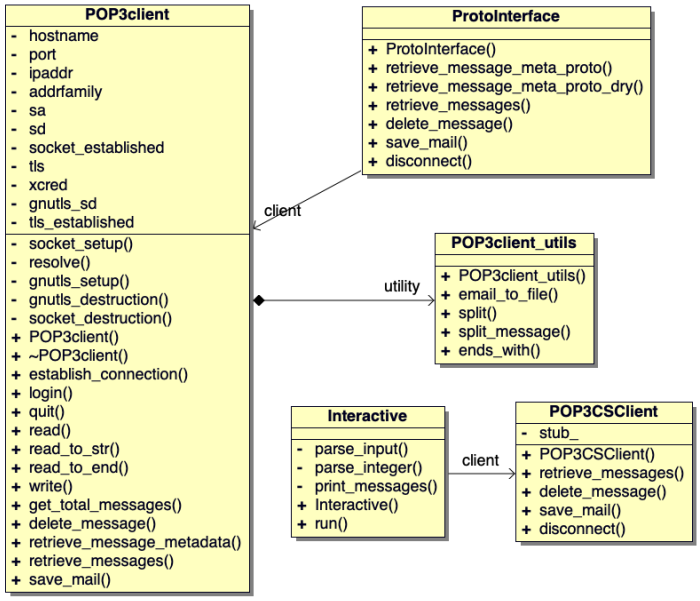

Figure 1: Klassendiagramm

2.1 Klassen

2.1.1 POP3client

Die POP3client Klasse umfasst die Funktionen, die es dem Programm erlauben, eine Verbindung zu einem POP3 Server aufzubauen und dessen Daten auszuwerten.

2.1.2 POP3client_utils

Die POP3client_utils Klasse umfasst Helfer-Funktionen, die die POP3Client Klasse unterstützen.

2.1.3 ProtoInterface

Die ProtoInterface Klasse verarbeitet Daten der POP3Client Klasse und konvertiert diese zu Protobuf Strukturen.

2.1.4 POP3CSClient

Die POP3CSClient Klasse ist eine simple gRPC Client Klasse, die Daten des gRPC Servers empfängt und verarbeitet.

2.1.5 POP3CSImplementation

Die POP3CSImplementation Klasse umfasst einen simplen gRPC Server.

2.1.6 Interactive

Die Interactive Klasse erlaubt die dynamische Eingabe von Befehlen während der Runtime des Programmes.

3 Implementierung

Für Variablen sowie Funktionen wurde "Snake case" benutzt. Sämtliche vorgegebene Codierungskonventionen wurden nach Möglichkeit eingehalten. Um sämtliche Teile des Programmes zu vereinfachen, wurden mehrere externe Bibliotheken verwendet, diese sind in Sektion 4.3 kurz beschrieben.

3.1 POP3 Protokoll

Das Post Office Protocol (POP) ist ein Übertragungsprotokoll, mit dem Clients die Möglichkeit haben, E-Mails von einem E-Mail Server herunterzuladen. Version 3 (POP3) des Protokolls wird im RFC 1939 [6] beschrieben.

3.1.1 POP3 Commands

USER [**username**] Erster Login-Command. Erlaubt es dem Client, einen User anzugeben.

PASS [**password**] Zweiter Login-Command. Erlaubt es dem Client, ein Passwort für den User anzugeben.

STAT Gibt die Anzahl von E-Mails und deren totale Größe zurück.

LIST Gibt die einzelnen E-Mails und deren Größe zurück.

RETR [**message**] Gibt ein gesammtes E-Mail zurück.

TOP [**message**] Gibt die Metadaten eines E-Mails zurück.

DELE [**message**] Löscht ein E-Mail.

RSET [**message**] Macht das Löschen eines E-Mails rückgängig.

NOOP POP3 Server gibt eine Nachricht zurück. Dient nur als Test-Command.

QUIT Beendet die Session und speichert getroffene Änderungen.

4 Kurze Beschreibung diverser Codeblöcke

4.1 POP3

4.1.1 Auflösen einer Domain

Die Verwendung der asio Bibliothek war in diesem Projekt leider nicht möglich, daher werden die Domains mit Funktionen aus der netdb.h C Bibliothek aufgelöst.

```
int POP3client::resolve() {
  hostent *rh = gethostbyname(hostname.c_str());
  if(rh == NULL){
      return 1;
  }

  in_addr **ipptr = (struct in_addr**)rh->h_addr_list;

  addrfamily  = rh->h_addrtype;
  ipaddr      = inet_ntoa(*ipptr[0]);
  return 0;
}
```

Source Code 1: Auflösen einer Domain ohne asio

4.1.2 Erstellen einer C Socket

Da die asio Bibliothek nur TLS über OpenSSL unterstützt, wurde eine normale C Socket implementiert.

```
int POP3client::socket_setup(){
  int err;

  sd = socket(addrfamily, SOCK_STREAM, 0);
  memset(&sa, '\0', sizeof(sa));

  sa.sin_family = addrfamily;
  sa.sin_port = htons(port);

  inet_pton(addrfamily, ipaddr.c_str(), &sa.sin_addr);
  err = connect(sd, (struct sockaddr *) &sa, sizeof(sa));

  socket_established = true;
  return 0;
}
```

Source Code 2: Erstellen einer C Socket

4.1.3 Implementation von GnuTLS

Die Funktion aus dem Snippet 3 zeigt, wie die Socket aus 2 benutzt wird, um einen TLS Handshake zu initialisieren.

```
int POP3client::gnutls_setup(){
  // X.509 authentication is used
  gnutls_global_init();
  gnutls_certificate_allocate_credentials(&xcred);
  gnutls_certificate_set_x509_system_trust(xcred);

  // link socket descriptor to gnutls socket descriptor
  gnutls_transport_set_int(gnutls_sd, sd);

  // initialize handshake
  int err_handshake = gnutls_handshake(gnutls_sd);

  if(err_handshake){
      return 1;
  }

  return 0;
}
```

Source Code 3: Gekürzte GnuTLS Setup Funktion

4.1.4 Lesen aus dem Socket

Da der Großteil der POP3 Funktionen mit einem Punkt, gefolgt von einem Absatz endet, ist es möglich, größere Nachrichten zu lesen, indem man nach diesem End-Signal sucht.

```
string POP3client::read_to_end(){
  char buff[8000]{};
  memset(buff, 0, sizeof(buff));

  string key = ".\r\n";
  string rec = "";
  string tmp = "";
```

```
  while(!utility.ends_with(tmp, key)){
      if(tls){
          gnutls_record_recv(gnutls_sd, buff, sizeof(buff));
      } else {
          recv(sd, buff, sizeof(buff), 0);
      }
      tmp = buff;
      memset(buff, 0, sizeof(buff));

      rec += tmp;
  }
  return rec;
}
```

Source Code 4: Funktion, die aus dem Socket liest, bis der Key erreicht wird

4.1.5 Schreiben in das Socket

Um in das Socket zu schreiben, werden Strings, die den Befehl enthalten, in Char Arrays umgewandelt. Diese werden, basierend darauf ob TLS verwendet wird, mit der dementsprechenden Funktion in das Socket geschrieben.

```
int POP3client::write(std::string msg){
  const char *cmsg = msg.c_str();
  int msg_len = strlen(cmsg);
  int result = 0;
  if(tls){
      result = gnutls_record_send(gnutls_sd, cmsg, msg_len);
  } else {
      result = send(sd, cmsg, msg_len, 0);
  }

  return result;
}
```

Source Code 5: Funktion, die in das Socket schreibt

4.2 Protobuf und gRPC

4.2.1 Protobuf

Protobuf [4] wurde implementiert, um die Verwendung von gRPC für den Austausch von Daten zu ermöglichen.

```
# Message Metadaten
message MailMeta {
  optional int32 message_id = 1;
  optional string from = 2;
  optional string subject = 3;
  optional string date = 4;
}

# Liste von Message Metadaten
message MailList {
  repeated MailMeta mails = 1;
}
```

Source Code 6: Protobuf Klassen für E-Mail Metadaten sowie eine Liste von diesen

4.2.2 gRPC

gRPC [5], auch bekannt als gRPC Remote Procedure Calls, wird benutzt als RPC System. gRPC benutzt HTTP/2 für den Transport und Protocol Buffer als die "interface description language."

```
service POP3CS {
  rpc get_mail_list(Operation) returns (MailList) {}

  rpc delete_message(Operation) returns (Success) {}
  rpc save_mail(Operation) returns (Success) {}

  rpc disconnect(Operation) returns (Success) {}
}
```

Source Code 7: gRPC Routen

Die POP3CSClient Klasse benutzt einen gRPC Client, um Daten des POP3 Servers über die POP3CSImplementation Klasse anzufragen.

4.3 Externe Bibliotheken

4.3.1 CLI11

CLI11 [1] ermöglicht eine einfache Verarbeitung von Kommandozeilenargumenten mit eingebauten Methoden zum Überprüfen der angegebenen Werte. Auf diese Argumente wird in Sektion 5 näher eingegangen.

4.3.2 tabulate

Um die E-Mails in einer optisch ansprechenden Art darzustellen, wird eine Tabelle mittels tabulate [10] erstellt.

```
bubble> list 3
+------------+----------------------+--------------------------+
| Message ID | Recieved From        | Subject                  |
+------------+----------------------+--------------------------+
| 6          | David Fischer <...>  | Linkedin ...             |
+------------+----------------------+--------------------------+
| 5          | David Fischer <...>  | Sie werden wahrgenommen! |
+------------+----------------------+--------------------------+
| 4          | David Fischer <...>  | DJI MSDK iOS Delay Notice |
+------------+----------------------+--------------------------+
```

Source Code 8: Ausgabe des *list* Befehls

4.3.3 spdlog

Um Informationen zu loggen, wird die spdlog [8] Bibliothek verwendet. Die *FileSink* wird für genauere Details benutzt, während die *ConsoleSink* benutzt wird, um die Arbeitsweise des Programms zu demonstrieren.

```
auto console = spdlog::stdout_color_mt("console");
auto logger = spdlog::basic_logger_mt("logger", "logs/basic-log.txt");
```

Source Code 9: Erstellen von spdlog Console und File Sinks

4.3.4 JSON

Um, wie in Sektion 5 angemerkt, die Konfiguration aus einem JSON File zu lesen, wird die JSON for modern C++ [7] verwendet.

4.3.5 httplib

Neben dem *bubble* executable wird ein executable namens *bubble_http* erstellt. Dieses öffnet einen httplib [11] HTTP Server auf Port 5001, der es dem Benutzer erlaubt, Mails online mittels gRPC zu verwalten.

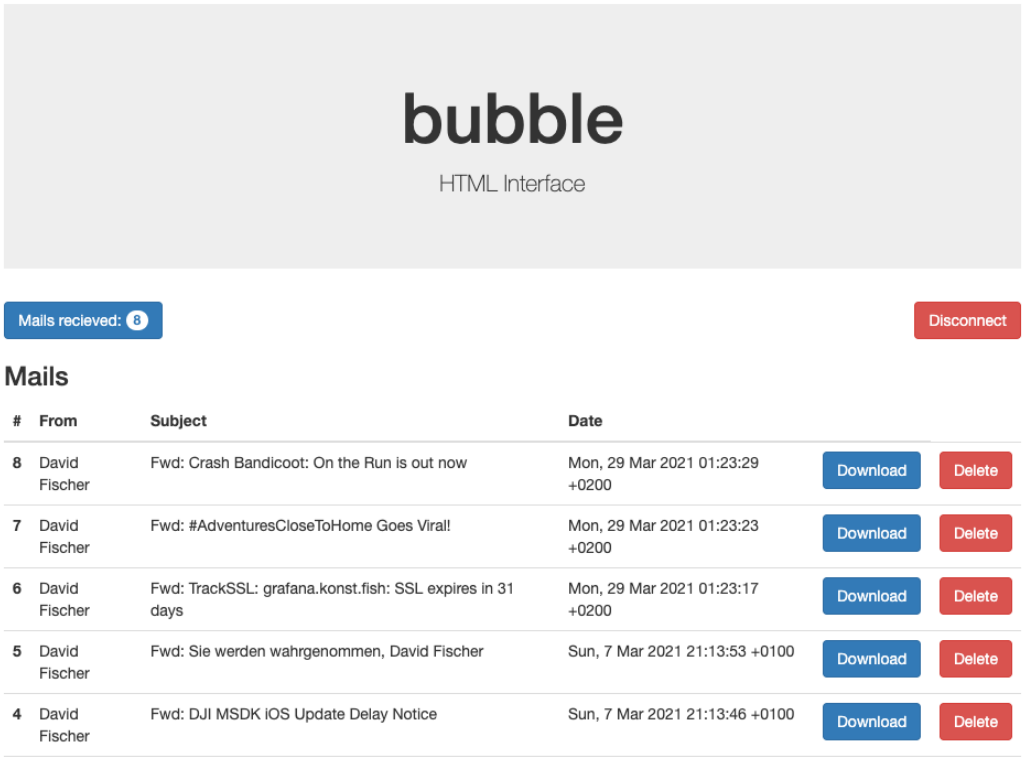

Figure 2: Bubble Web Interface

4.3.6 inja

Um die Web Interface präsentabel darstellen zu können, wird die inja Template Engine [9] verwendet. In Kombination mit httlib und gRPC kann die Web Interface dynamisch mit Daten befüllt werden.

4.3.7 GnuTLS

Um GnuTLS zu implementieren, wurde die C++ API verwendet [2]. Dies ist in Sektion 4.1.3 näher beschrieben.

5 Verwendung

5.1 Kommandozeilenargumente

5.1.1 Konfiguration

-s,–server Domain des POP3 Servers

-u,–user Username des Accounts.

-p,–pass Username des Accounts.

-t,–tls Verbindung mittels TLS etablieren.

–port Benutzerdefinierten Port angeben.

-j,–json Configuration aus einem JSON File lesen

5.1.2 Interaktive Eingabe von Befehlen

-i,–interactive Interaktive Eingabe von Befehlen aktivieren.

```
bubble> help
Bubble Interactive Client:
dl / download <message_id>  - download email with id
rm / delete <message_id>    - delete email with id
ls / list <amount>          - list amount of mails
exit                        - disconnects the session
help                        - displays this message
```

Source Code 10: Ausgabe des help Befehls des Interaktiven Clients

6 Projektstruktur

```
/
├── LICENSE
├── meson_options.txt
├── meson.build
├── README.md
├── RESEARCH.md
├── CHANGELOG.org
├── config.json
├── .gitignore
├── include
│   ├── Interactive.h
│   ├── POP3client.h
│   ├── ProtoInterface.h
│   └── Util.h
├── src
│   ├── Interactive.cpp
│   ├── main.cpp
│   ├── pop3.proto
│   ├── POP3cilent.cpp
│   ├── ProtoInterface.cpp
│   └── Util.cpp
├── doc
│   ├── dokumentation.tex
│   ├── references.bib
│   └── dokumentation.pdf
├── http_server
│   ├── include
│   │   └── POP3CSClient.h
│   └── src
│       ├── main.cpp
│       └── POP3CSClient.cpp
├── templates
│   └── render.html
└── build
```

References

[1] CLIUtils. Cli11. https://github.com/CLIUtils/CLI11, 2021.

[2] GnuTLS.org. Client example using the c++ api. https://gnutls.org/manual/html_node/Client-example-in-C_002b_002b.html, 2021.

[3] GnuTLS.org. The gnutls transport layer security library. https://gnutls.org/, 2021.

[4] Google. Protocol buffers. https://developers.google.com/protocol-buffers, 2021.

[5] gRPC Authors. A high performance, open source universal rpc framework. https://grpc.io/, 2021.

[6] M. Rose J. Myers, Carnegie Mellon. Post office protocol - version 3. https://tools.ietf.org/html/rfc1939, 1996.

[7] Niels Lohmann. Json for modern c++. https://nlohmann.github.io/json/, 2021.

[8] Gabi Melman. spdlog. https://github.com/gabime/spdlog, 2021.

[9] pantor. Inja is a template engine for modern c++. https://github.com/pantor/inja, 2020.

[10] Pranav. tabulate. https://github.com/p-ranav/tabulate, 2021.

[11] yhirose. cpp-httplib. https://github.com/yhirose/cpp-httplib, 2020.