МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»
(БГТУ им. В.Г.Шухова)

Лабораторная работа №7
дисциплина «Сети ЭВМ и телекомуникации»
по теме «Протоколы POP3 и SMTP»

Выполнил: студент группы ВТ-31 Макаров Д.С.
Проверил: Федотов Е.А.

Белгород 2020

Лабораторная работа №7

«Протоколы POP3 и SMTP»

Цель работы:изучить принципы и характеристику протоколов POP3 и SMTP и составить программу для приема/отправки электронной почты.

Вариант 6

Содержание отчета

1. Краткие теоретические сведения.
2. Основные функции API, использованные в данной работе.
3. Разработка программы. Блок-схемы программы.
4. Анализ функционирования разработанных программ.
5. Выводы.
6. Тексты программ. Скриншоты программ.

Ход работы

1. **Краткие теоретические сведения.**

POP3 (*Post Office Protocol Version 3*) - стандартный Интернет-протокол прикладного уровня, используемый клиентами электронной почты для извлечения электронного сообщения с удаленного сервера по TCP/IP-соединению. В некоторых небольших узлах Интернет бывает непрактично поддерживать систему передачи (*MTS - Message Transport System*). Рабочая станция может не иметь достаточных ресурсов для обеспечения непрерывной работы SMTP-сервера. Для "*домашних ЭВМ*" слишком дорого поддерживать связь с Интернет круглые сутки. Но доступ к электронной почте необходим как для таких малых узлов, так и индивидуальных ЭВМ. Для решения этой проблемы разработан протокол POP3 (Post Office Protocol - Version 3, STD- 53 M. Rose, RFC-1939). Этот протокол обеспечивает доступ узла к базовому почтовому серверу. POP3 не ставит целью предоставление широкого списка манипуляций с почтой. Почтовые сообщения принимаются почтовым сервером и сохраняются там, пока на рабочей станции клиента не будет запущено приложение POP3. Это приложение устанавливает соединение с сервером и сообщения оттуда. Почтовые сообщения на сервере стираются.

SMTP (Simple Mail Transfer Protocol) - широко используемый сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP. SMTP впервые был описан в RFC 821 (1982 год) последнее обновление в RFC 5321 (2008) включает масштабируемое расширение - ESMTP (Extended SMTP). В настоящее время под «протоколом SMTP», как правило, подразумевают и его расширения. Протокол SMTP предназначен для передачи исходящей почты, используя для этого порт TCP 25.

2. **Разработка программы.**

   В ходе работы было разработано консольное приложение на языке C++ которое может выполнять следующие действия:

   - подключение к почтовому серверу.
   - авторизация и аутентификация.
   - проверка входящих сообщений.
   - отправка сообщений с вложением.

3. **Анализ функционирования разработанных программ.**

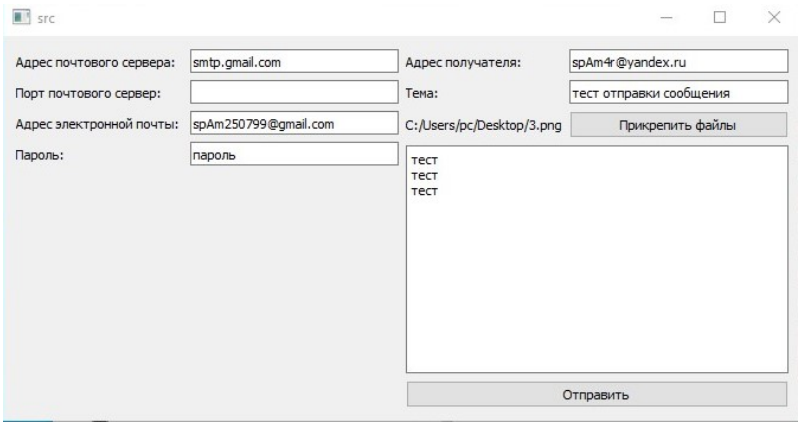

Рис. 1: Пример отправки сообщения с вложением

4. **Выводы.**

В данной лабораторной работе была реализована программа для взаимодействия с ARP таблицей при помощи библиотек Winsock и IP Helper.

5. **Тексты программ. Скриншоты программ.**

   Тексты программ см. в приложении.

7. **Контрольные вопросы**

   - Что представляет собой протокол POP3? С какой целью он был разработан?

     Стандартный интернет-протокол прикладного уровня, используемый клиентами электронной почты для получения почты с удалённого сервера по TCP-соединению.

   - Опишите процесс работы протокола POP3.

     Перед работой через протокол POP3 сервер прослушивает порт 110. У POP3 сервера может быть INACTIVITY AUTOLOGOUT таймер. Этот

таймер должен быт, по крайней мере, с интервалом 10 минут. Это значит, что если клиент и сервер не взаимодействуют друг с другом, сервер автоматически прерывает соединение и при этом не переходит в режим UPDATE.

- Формат команд протокола POP3.

  Команды POP3 состоят из ключевых слов, за некоторыми следует один или более аргументов. Ключевые слова и аргументы состоят из печатаемых ASCII символов. Ключевое слово и аргументы разделены одиночным пробелом. Ключевое слово состоит от 3-х до 4-х символов, а аргумент может быть длиной до 40-ка символов.

  Ответы в POP3 состоят из индикатора состояния и ключевого слова, за которым может следовать дополнительная информация. Существует только два индикатора состояния: "+OK" - положительный и "-ERR" - отрицательный. Ответы на некоторые команды могут состоять из нескольких строк.

- Из каких частей состоит POP3-сессия?

  OP3 сессия состоит из нескольких режимов. Как только соединение с сервером было установлено и сервер отправил приглашение, то сессия переходит в режим AUTHORIZATION (Авторизация). В этом режиме клиент должен идентифицировать себя на сервере. После успешной идентификации сессия переходит в режим TRANSACTION (Передача). В этом режиме клиент запрашивает сервер выполнить определённые команды. Когда клиент отправляет команду QUIT, сессия переходит в режим UPDATE. В этом режиме POP3 сервер освобождает все занятые ресурсы и завершает работу. После этого TCP соединение закрывается.

3. Как осуществляется взаимодействие SMTP и POP3?

Конструкция протокола POP3 обеспечивает возможность пользователю обратиться к своему почтовому серверу и изъять накопившуюся для него почту. Пользователь может получить доступ к POP3-серверу из любой точки доступа к Internet. При этом он должен запустить специальный почтовый агент, работающий по протоколу POP3, и настроить его для работы со своим почтовым сервером. Сообщения доставляются клиенту по протоколу POP3, а посылаются при помощи SMTP. То есть на компьютере пользователя существуют два отдельных агента-интерфейса к почтовой системе – доставки (POP3) и отправки (SMTP).

4. Минимальный набор команд и порядок их применения для отправки почты по протоколу SMTP.

```
S: (ожидает соединения)
```

```
C: (Подключается к порту 25 сервера)
S:220 mail.company.tld ESMTP is glad to see you!
C:HELO
S:250 domain name should be qualified
C:MAIL FROM: <someusername@somecompany.ru>
S:250 someusername@somecompany.ru sender accepted
C:RCPT TO: <user1@company.tld>
S:250 user1@company.tld ok
C:DATA
S:354 Enter mail, end with "." on a line by itself
C:From: Some User <someusername@somecompany.ru>
C:To: User1 <user1@company.tld>
C:Subject: tema
C:Content-Type: text/plain
C:
C:Hi!
C:.
S:250 769947 message accepted for delivery
C:QUIT
S:221 mail.company.tld CommuniGate Pro SMTP closing connection
S: (закрывает соединение)
```

Приложение

Содержимое файла mailclient.cpp

```
#include "mail_client.h"

MailClient::MailClient(){
    connect(&loginButton,SIGNAL(pressed()),this,SLOT(loginButtonPush()));
    connect(&updateButton,SIGNAL(pressed()),this,SLOT(updateButtonPush()));
    connect(&deleteButton,SIGNAL(pressed()),this,SLOT(delButtonPush()));

    ↪  connect(&mailList,SIGNAL(itemClicked(QListWidgetItem)),this,SLOT(listItemClicked(QListWidgetIte
    generalLayout.addLayout(&configLayout);
        configLayout.addLayout(&formLayout);
            serverAdressLabel.setText("Адрес почтового сервера: ");
            adressLabel.setText("Адрес электронной почты: ");
            passwordLabel.setText("Пароль: ");

            formLayout.addRow(&serverAdressLabel,&serverAdressLine);
            formLayout.addRow(&adressLabel,&adressLine);
            formLayout.addRow(&passwordLabel,&passwordLine);
        loginButton.setText("Подключиться");
        configLayout.addWidget(&loginButton);
    generalLayout.addLayout(&emailListLayout);
        updateButton.setText("Обновить список");
        emailListLayout.addWidget(&mailList);
        emailListLayout.addWidget(&updateButton);
    generalLayout.addLayout(&emailContentLayout);
        deleteButton.setText("Удалить сообщение");
        emailContentLayout.addWidget(&emailContent);
        emailContentLayout.addWidget(&deleteButton);
    this->setLayout(&generalLayout);
};


void MailClient::loginButtonPush(){
    QTextStream log_stream(&log);
    QString request;
    QString response;
    if (this->connected){
        request = "QUIT\r\n";
        socket.write(request.toLocal8Bit());
        socket.waitForBytesWritten();
        qDebug() << "[Client]: " << request;
        log_stream << "[Client]: " << request;
        socket.waitForReadyRead();
        response = socket.readAll();
        qDebug() << "[Server]: " << response;
        log_stream << "[Server]: " << response;
        socket.close();
        loginButton.setText("Подключение");
        this->connected = false;
        QMessageBox msgBox;
        msgBox.setText("Лог сессии");
        msgBox.setDetailedText(log);msgBox.setStandardButtons(QMessageBox::Ok);
        msgBox.setDefaultButton(QMessageBox::Ok);
        msgBox.exec();
    }else{
        qDebug() << "Connecting...";
        socket.connectToHost(serverAdressLine.text(), 110);
```

```
        if(socket.waitForConnected(5000)){
            qDebug() << "Connected!";
            socket.waitForReadyRead();
            response = socket.readAll();
            qDebug() << "[Server]: " << response;
            log_stream << "[Server]: " << response;

                request = "USER " + adressLine.text() + "\r\n";
                socket.write(request.toLocal8Bit());
                socket.waitForBytesWritten();
                qDebug() << "[Client]: " << request;
                log_stream << "[Client]: " << request;

                socket.waitForReadyRead();
                response = socket.readAll();
                qDebug() << "[Server]: " << response;
                log_stream << "[Server]: " << response;

                request = "PASS " + passwordLine.text() + "\r\n";
                socket.write(request.toLocal8Bit());
                socket.waitForBytesWritten();
                qDebug() << "[Client]: " << request;
                log_stream << "[Client]: " << request;

                socket.waitForReadyRead();
                response = socket.readAll();
                qDebug() << "[Server]: " << response;
                log_stream << "[Server]: " << response;

                loginButton.setText("Отключиться");
                this->connected = true;

            }else{
                qDebug() << "Not connected!";
            }
        }
};
void MailClient::updateButtonPush(){
    mailList.clear();
    if (!connected){
        return;
    }
    QTextStream log_stream(&log);
    QString request;
    QString response;

    request = "LIST\r\n";
    socket.write(request.toLocal8Bit());
    socket.waitForBytesWritten();
    qDebug() << "[Client]: " << request;
    log_stream << "[Client]: " << request;

    socket.waitForReadyRead();
//    qDebug() << "Ready to receive " << socket.bytesAvailable() << " bytes\n";
    response = socket.readAll();
    qDebug() << "[Server]: " << response;
    log_stream << "[Server]: " << response;

    int messages = response.split(" ")[1].toInt();
    QStringList items;
```

```
    while (items.length() < messages){
        socket.waitForReadyRead();
        response = socket.readAll();
        qDebug() << "[Server]: " << response;
        log_stream << "[Server]: " << response;
        items.append(response.split("\r\n"));
    }
    for (auto item: items){
        if (item != "." && item != ""){
            mailList.addItem(item + " байт");
        }
    }
    response = socket.readAll();
    qDebug() << "[Server]: " << response;
    log_stream << "[Server]: " << response;
};
void MailClient::delButtonPush(){
    QTextStream log_stream(&log);
    QString request;
    QString response;
    int index = curr_msg;
    request = "DELE " + QString::number(index) + "\r\n";
    socket.write(request.toLocal8Bit());
    socket.waitForBytesWritten();
    qDebug() << "[Client]: " << request;
    log_stream << "[Client]: " << request;

    socket.waitForReadyRead();
//    qDebug() << "Ready to receive " << socket.bytesAvailable() << " bytes\n";
    response = socket.readAll();
    qDebug() << "[Server]: " << response;
    log_stream << "[Server]: " << response;
    curr_msg = 0;
};
void MailClient::listItemClicked(QListWidgetItem *item){
    QTextStream log_stream(&log);
    QString request;
    QString response;
    int index = item->text().split(" ")[0].toInt();
    curr_msg = index;
    request = "RETR " + QString::number(index) + "\r\n";
    socket.write(request.toLocal8Bit());
    socket.waitForBytesWritten();
    qDebug() << "[Client]: " << request;
    log_stream << "[Client]: " << request;

    socket.waitForReadyRead();
    response = socket.readAll();
    qDebug() << "[Server]: " << response;
    log_stream << "[Server]: " << response;

    QString message;
    QTextStream message_stream(&message);
    int read = 0;
    int all = response.split(" ")[1].toInt();
    while (socket.waitForReadyRead(1000)){
        read += socket.bytesAvailable();
        response = socket.readAll();
        qDebug() << read << "/" << all;
        message_stream << response;
```

```
    }

    emailContent.setText(message);
};
```

Содержимое файла mailclient.h

```
#pragma once
#include <QtWidgets>

#include <QFileDialog>
#include <QDebug>
#include <QMessageBox>
#include <QTcpSocket>

class MailClient : public QWidget{
    Q_OBJECT
    public:
        MailClient();
    private:
        QString log;
        bool connected;
        QTcpSocket socket;
        int curr_msg = 0;

        QHBoxLayout generalLayout;
            QVBoxLayout configLayout;
                QFormLayout formLayout;
                    QLabel passwordLabel,adressLabel,serverAdressLabel;
                    QLineEdit passwordLine,adressLine,serverAdressLine;
                QPushButton loginButton;
            QVBoxLayout emailListLayout;
                QListWidget mailList;
                QPushButton updateButton;
            QVBoxLayout emailContentLayout;
                QTextEdit emailContent;
                QPushButton deleteButton;
    public slots:
        void loginButtonPush();
        void updateButtonPush();
        void delButtonPush();
        void listItemClicked(QListWidgetItem *item);
};
```

Содержимое файла main.cpp

```
#include <QApplication>
#include "mail_client.h"
int main(int argc, char *argv[]){
    QApplication app(argc,argv);
    MailClient client;
    client.show();
    return app.exec();
}
```