Laboratory Work 4

Network Programming

Email protocols: SMTP and POP3

FAF 141 student:
Ana-Maria Brinza

Supervisor :
Dumitru Ciorba

Chisinau 2017

1 General Theory

Email is delivered using a client/server architecture. An email message is created using a mail client program. This program then sends the message to a server. The server then forwards the message to the recipient's email server, where the message is then supplied to the recipient's email client.

To enable this process, a variety of standard network protocols allow different machines, often running different operating systems and using different email programs, to send and receive email.

The following protocols discussed are the most commonly used in the transfer of email.

Mail Transport Protocols: SMTP

Mail delivery from a client application to the server, and from an originating server to the destination server, is handled by the Simple Mail Transfer Protocol (SMTP) .

The primary purpose of SMTP is to transfer email between mail servers. To send email, the client sends the message to an outgoing mail server, which in turn contacts the destination mail server for delivery. For this reason, it is necessary to specify an SMTP server when configuring an email client. One important point to make about the SMTP protocol is that it does not require authentication. This allows anyone on the Internet to send email to anyone else or even to large groups of people. It is this characteristic of SMTP that makes junk email or spam possible. Modern SMTP servers attempt to minimize this behavior by allowing only known hosts access to the SMTP server.

Mail Access Protocols: POP3

There are two primary protocols used by email client applications to retrieve email from mail servers: the Post Office Protocol (POP) and the Internet Message Access Protocol (IMAP).

Unlike SMTP, both of these protocols require connecting clients to authenticate using a username and password. By default, passwords for both protocols are passed over the network unencrypted.

When using a POP server, email messages are downloaded by email client applications. By default, most POP email clients are automatically configured to delete the message on the email server after it has been successfully transferred, however this setting usually can be

changed.

POP is fully compatible with important Internet messaging standards, such as Multipurpose Internet Mail Extensions (MIME), which allow for email attachments.

POP works best for users who have one system on which to read email. It also works well for users who do not have a persistent connection to the Internet or the network containing the mail server. Unfortunately for those with slow network connections, POP requires client programs upon authentication to download the entire content of each message. This can take a long time if any messages have large attachments.

The most current version of the standard POP protocol is POP3.

2 Task for the laboratory work

The main purpose of this laboratory work is to understand how the emailing system works and to analyze two main protocols: SMTP and POP3; The aim is to create a Mail application that will allow one simple user/client to send and receive messages.

So, mainly what we have to do is to create a simple Mail client able to send and receive messages through an email account. Additionally, the user would be able to send a letter with an attachment.

3 The implementation of the task

In order to implement this task, I used the Apache Commons Email library. This library simplifies a lot your work and it is perfect in case when we want to send the message.

Some of the most important classes provided by it, would be:

SimpleEmail used for sending simple text messages. I used this class in my program.

```
Email email = new SimpleEmail();
email.setHostName("127.0.0.1");
email.setSmtpPort(25);
email.setFrom("b@mail.md");
email.setSubject("Mesaj de la mine pentru tine");
email.setMsg("Mesaj frumos si fragut!");
email.addTo("ana@mail.md");
email.send();
```

MultiPartEmail a class that is used to send composed messages, that include attachments, for example. I implemented it in my program.

```
// Create the attachment
EmailAttachment attachment = new EmailAttachment();
attachment.setPath("D:\\UTM\\Anul 3\\PR\\Lab#4\\pr_lab4/UTM.png");
attachment.setDisposition(EmailAttachment.ATTACHMENT);
attachment.setDescription("Picture of John");
attachment.setName("John");

// Create the email message
MultiPartEmail email = new MultiPartEmail();
email.setHostName("127.0.0.1");
email.setFrom("b@mail.md");
email.setSubject("The picture");
email.setMsg("Here is the picture you wanted");
email.addTo("ana@mail.md");


// add the attachment
```

```
email.attach(attachment);

// send the email
email.send();
```

In order to read the messages I used the class POP3Client of the same Apache library. The methods of this class allow us to log in, to list the messages, to obtain a message and to interrupt the working process. All these are possible because of a TCP connection to the port 110.

The implementation that I used in my program is the following:

```
POP3Client client = new POP3Client();
client.connect( "127.0.0.1", 110 );
if( client.login( "ana", "ana" ) )

{

    POP3MessageInfo[] messages = client.listMessages();
    System.out.println( "Mesaje: " + messages.length );
    System.out.println( "Textul mesajului");
    Reader r = client.retrieveMessage( messages[ 7 ].number );
    BufferedReader br = new BufferedReader( r );
    String line;
    while( ( line = br.readLine()) != null )
    {
        System.out.println(line);
    }
}
else
{
    System.out.println( "Logare fara succes..." );
}
client.logout();
client.disconnect();
```

In the end, after running my program, I get the following:

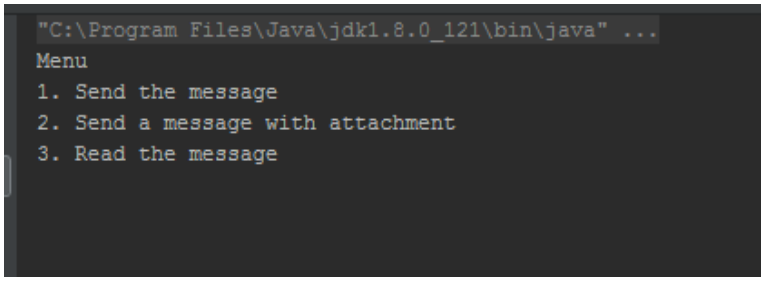

Fig. 1 *The result after running the program*

So, in case we choose the opption one:

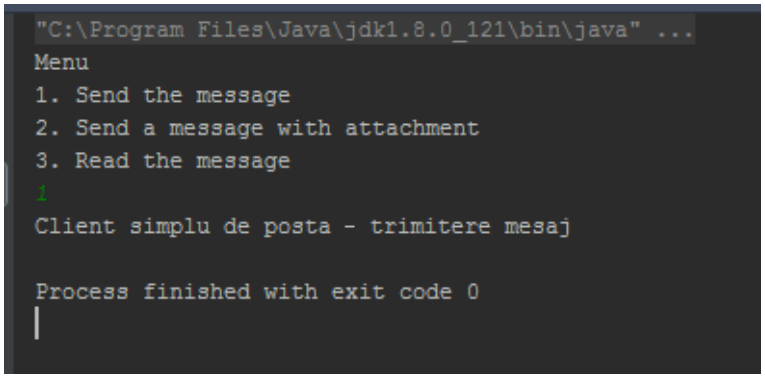

Fig. 2 *Choosing option one: Send a simple text message to our user*

Now, if we want to check if it indeed was executed, we can go to our directory *C:* and check all the received mails of the specified user, in our case it is *ana*.

Also, we could choose the option 3 from our menu and check the message that was sent. The result is in Fig. 3

In case we send an image with an attachment, the user will get another kind of response. The result can be seen in the Fig. 4

```
"C:\Program Files\Java\jdk1.8.0_121\bin\java" ...
Menu
1. Send the message
2. Send a message with attachment
3. Read the message
3
Client simplu de posta - citire mesaj
Mesaje: 9
Textul mesajului
x-sender: <b@mail.md>
x-receiver: <ana@mail.md>
Received: from DESKTOP-THHTSTK ([127.0.0.1]) by gmail.com with Quick 'n Easy Mail Server SMTP (1.0.0.0);
    Wed, 24 May 2017 15:56:41 GMT
Date: Wed, 24 May 2017 18:56:40 +0300 (EEST)
From: "b@mail.md" <b@mail.md>
To: "ana@mail.md" <ana@mail.md>
Message-ID: <1637070917.0.1495641401032.JavaMail.Amy@DESKTOP-THHTSTK>
Subject: Mesaj de la mine pentru tine
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Mesaj frumos si fragut!

Process finished with exit code 0
```

Fig. 3 *Result of a simple text message*

```
3
Client simplu de posta - citire mesaj
Mesaje: 9
Textul mesajului
x-sender: <b@mail.md>
x-receiver: <ana@mail.md>
Received: from DESKTOP-THHTSTK ([127.0.0.1]) by gmail.com with Quick 'n Easy Mail Server SMTP (1.0.0.0);
    Wed, 24 May 2017 16:57:59 GMT
Date: Wed, 24 May 2017 19:57:59 +0300 (EEST)
From: "b@mail.md" <b@mail.md>
To: "ana@mail.md" <ana@mail.md>
Message-ID: <625576447.1.1495645079539.JavaMail.Amy@DESKTOP-THHTSTK>
Subject: The picture
MIME-Version: 1.0
Content-Type: multipart/mixed;
    boundary="----=_Part_0_1044036744.1495645079423"

------=_Part_0_1044036744.1495645079423
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Here is the picture you wanted
------=_Part_0_1044036744.1495645079423
Content-Type: application/octet-stream; name=John
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=John
Content-Description: Picture of John

iVBORw0KGgoAAAANSUhEUgAAAu4AAAC7CAMAAAANH+uVAAAAGXRFWHRTb2Z0d2FyZQBBZG9iZSBJ
bWFnZVJlYWR5ccllPAAAABhQTFRF+vz9MnagA1aKp8TWdKK/4uvxyNrm////NyRFeQAAAAh0Uk5T
//////////wDeg71ZAAAVj0lEQVR42uyd2YKjOAxFZbzw/3/cVayWLNmGkGpIrh5mupJgtoPQZotG
CORrhHAJIMAdAgHuEAhwh0CAOwQC3CEQ4A6BAHcIBLhDIHfAndIfSCRrr8SP4Zrzi0kfHsri23Gn
MPyFkLXXmUe3/hmuOMF9ODcN19Y/PQD6ctzjn9AuQSPxHOzPXLzgnLzYbeL0Q75Yu98C9/2hC9cr
9x33AbgD9xvgXhB6zQtreXaAO3C/Fe6F/fGKFJYRcAfu98L9QvVeDgXcgfu9cL/QWS1fFMAduN8M
93SVs6g8J4A7cC/8urdKaOF+mTWjBHmAO3BfB4x/Iy3cL3NWFasIuAP3/3saJYAXgXdtGOAO3O+G
```

Fig. 4 *Result of a message with attachment*

4 Conclusion

This laboratory work gave me the chance to analyze how actually an email is delivered and to learn more about it's client/server architecture.

I understood how important are the SMTP and POP3 protocols in the process of transferring a mail. You can not send a mail without having a mail transport protocol, one simple example to implement is SMTP, and you can not to receive a mail without a mail access protocol, such as POP3 - wchich is actual nowadays and one of the most used.

5 Appendix

1. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/3/html/Reference_Guide/ch-email.html
2. https://commons.apache.org/proper/commons-email/userguide.html
3. https://github.com/anamariabrinza/PRlabs/tree/master/Lab%234/pr_lab4