1 Introduction

A request for comments (RFC) is a memorandum published by the Internet Engineering Task Force (IETF) describing methods, behaviors, research, or innovations applicable to the working of the Internet and Internet-connected systems.

RFC 821 contains the Simple Mail Transfer Protocol (SMTP). SMTP is used to send emails through the internet. Servers on the internet that handle email are running an SMTP server program. When you send an email, you have to specify one of these SMTP servers.

In this assignment you are going to write a so called forwarding SMTP server. You may do this in Java or C. For Java, a framework is provided.

2 SMTP Server

Your SMTP server has to accept incoming connections on a port that you can specify on startup of the program. The SMTP sequence as described in RFC 821 should be used. You only have to implement the commands HELO, MAIL, RCPT, DATA, NOOP, RSET and QUIT.

2.1 Handling of the Commands

2.1.1 HELO <hostname>

The server should respond to this command with 250 <serverhostname>. <hostname> may not be empty. In that case your server should respond with 501 Syntax: HELO hostname The HELO command may only be send once in a session. The appropriate response to more than one HELO is 503 Duplicate HELO.

This command needs no previous commands.

2.1.2 MAIL FROM: <sender>

<sender> may not be empty.

This command needs HELO to be executed first.
2.1.3 RCPT TO: <receiver>

receiver> may not be empty. Use the correct error message. Use the correct message to respond on success.

The command can be executed multiple times to specify multiple receivers.

The commands HELO and MAIL should be executed before this one.

2.1.4 DATA

DATA is a special command. You have to read several lines here until you see a line with a single dot on it, to be exactly the sequence <CR><LF>.<CR><LF> implies the ending of the data part. CR stands for Carriage Return and LF for Linefeed. Use the correct message to respond on success.

The command is also special since it implies the end of an email message. Now, the message should be stored and the state just after the execution of the HELO is restored, so that the server is ready for another message.

The commands HELO, MAIL and RCPT should precede this command.

2.1.5 NOOP

NOOP does nothing. The server should always replies with 250 Ok.

This command can always be executed and does not affect state.

2.1.6 RSET

This command can be executed at any time after the HELO, and restores the state to just after the execution of the HELO. The message that was started should be discarded. Reply with the correct message.

The command HELO should precede this command.

2.1.7 QUIT

QUIT closes the connection. We should respond properly and terminate the session appropriately. The server should be ready to accept an other connection.

2.2 State

As you hopefully have noticed, the commands need to be executed in a certain order. To check this order, we have to maintain state. Think of a good way to do so.

2.3 Sockets

Communication between multiple machines via a network is usually implemented with sockets. In Java, there are two types of sockets. The normal sockets, which create a connection, and the serverSockets which waits for a client to connect. A good starting point to read and learn more about sockets is http://java.sun.com/docs/books/tutorial/networking/sockets/.
2.4 Telnet

To get familiar with socket connections and SMTP, you can use a python dummy mail server and telnet to connect to this.
An example of playing with an SMTP server:

- Start a Terminal.
- Type `python -m smtpd -n -c DebuggingServer localhost:1025`
  This will start an smtp server on the local machine using port 1025.
- Start another Terminal, and keep this one next to your python server’s window.
- In the second terminal type `telnet localhost 1025`
  This will connect to your mail server and hopefully it greets you.
- In the telnet window you can type the command for the mail server.

For example:

```
(bakkerr@ow151) telnet localhost 1025
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 ow151.science.uva.nl Python SMTP proxy version 0.2
HELO ow151.science.uva.nl
250 ow151.science.uva.nl
NOOP
250 0k
NOOP
250 0k
RESET
502 Error: command "RESET" not implemented
RSET
250 0k
MAIL FROM: bakkerr@science.uva.nl
250 0k
RCPT TO: bakkerr@uva.nl
250 0k
etc, etc, etc.
```

2.5 Sending to an external SMTP server

Besides receiving messages, your server needs to send the received messages again to an external SMTP server. To do so, we again need a socket connection to that server, and now we are the client and send the commands.

2.6 Framework

Writing an SMTP server is quite a lot of work, and certainly when you have limited programming experience. To give you something to start with, you can
use the framework that is available. The framework is a minimal implementa-
tion of your SMTP server. It writes and listens to the terminal instead of a
socket connection. A Makefile is provided with the framework. To compile the
project, you just type make. And to run the server you use the command java
SMTPserver <listenport> <external_host> <external_port>.

For example:

(bakkerr@ow151) make
javac -g -Xlint SMTPmessage.java
javac -g -Xlint SMTPincoming.java
javac -g -Xlint SMTPoutgoing.java
javac -g -Xlint SMTPserver.java
(bakkerr@ow151) java SMTPserver 25 ow152.science.uva.nl 25
Starting SMTP server on ow151.science.uva.nl:25
Waiting for connection on port 25.
NOOP
Got message: NOOP
250 Ok
Adding message
0 bakkerr@science.uva.nl

^C(bakkerr@ow151)

The framework consists of four java classes.

2.6.1 SMTPmessage.java

In this file, you are going to make a datastructure for a SMTP message. All
variables should be private and you should make public getters and setters for
each variable that the program should be able to set or retrieve its value. An
example is given for the sender address.

2.6.2 SMTPincoming.java

This class handles the incoming connections. It implements Runnable, so that
it can be a thread. Threads are used when two or more parts of a program
should run at the same time. Search for some more information about threads.
You do not have to understand it all at this moment.

The global implementation of this class is already there. You have to modify
it such that it handles communication through sockets instead of the terminal,
and the message handling function needs to be extended and coupled to the
datastructure.

2.6.3 SMTPoutgoing.java

This class handles the outgoing connections. It also implements Runnable and
therefore is the second thread in this program. This thread should make a
connection to the external SMTP server each 10 seconds, but only when there
are messages to be send.

Again, a lot is available in this class, but there is certainly also a lot to add.
2.6.4 SMTPserver.java

This class contains the main function and creates the two threads.

The class does not need modifications, but you might extend the functionality a bit if you like.

3 What to hand in

This assignment consists of two parts, writing the SMTP server, and writing a small report.

3.1 Code

Of course you have to hand in your code. The code that you write should be commented in English or Dutch and be structured clearly. Use a consequent coding style. You should use the framework that is provided, but you may (and probably should) rewrite parts of it.

3.2 Report

As in any scientific assignment, you have to write a report about what, why and how you did this assignment. This report should be written in \LaTeX, and contains an introduction with description of the assignment, a section containing your implementation decisions, a section containing how your program should be used and some conclusions, results and remarks.

The report should be 3-4 pages A4 and written in English or Dutch.

In appendix A, which is not included in the page count, you should provide the output of a sample `telnet` run of your program using the following sequence of commands as input:

```
HELO
PRNT
HELO owxx.science.uva.nl
MAIL FROM: bakkerr@science.uva.nl
DATA
PRNT
RSET
MAIL FROM: youraddress@science.uva.nl
RCPT TO: r.bakker@uva.nl
RCPT TO: bakkerr@science.uva.nl
DATA
Hi Roy,

this is some test message to test my SMTP server.

kind regards,

You.

.
HELO
```
MAIL FROM: bakkerr@science.uva.nl
RCPT TO: youraddress@science.uva.nl
DATA
Hi there,

I'm glad you are testing your mail server.

Success with your assignment!

Roy.

The bones for this LaTeX report are also available.

4 Planning

For this assignment, you will have three weeks. The lab sessions in which you can work on the assignment are on Wednesdays from 14:00 till 17:00. The time to work on the assignment is approximately the time available in the labs times two, which in this case means six hours a week. You may work in groups of two, but working alone is also permitted. Attending the lab sessions is not obligatory, but certainly recommended. There is one global rule: When you attend the lab session, you may ask additional questions via email. When you can not make it to the labs, but might have questions, please send an email to me before the lab session. I will keep presentation lists.

The global weekly planning for this assignment:

4.1 Week one

Read the assignment and build a program that handles the commands as specified in this document or in the RFC. The commands are read through the socket connection, for example using telnet for input.

Finish: Monday April 6, 2009 23:59:59. If you send me your code before that time, I will have a look at it and discuss some point with you during the lab of Wednesday April 8. Note that it is not obligatory to submit anything before this date.

4.2 Week two

Create the datastructure and put all data generated by the commands in the datastructure. Also create a function that writes the current list of messages to the socket.

Finish: Monday April 13, 2009 23:59:59. If you send me your code before that time, I will have a look at it and discuss some point with you during the
lab of Wednesday April 15. Note that it is not obligatory to submit anything before this date.

4.3 Week three
Create the functionality that sends the messages to an external SMTP server. Test your code and write the report.


Note that the deadline is very strict. If, for some good reason, you can not hand in your assignment before the deadline, please contact me as soon as possible. We can find a way to solve the problem. This means that an email five minutes before the deadline will not result in extension of the deadline, but a few days before might.

The assignment should be submitted via blackboard.

5 Bonus:
To earn some more points, you can implement one of these additional features. Maybe I will think of some more bonus assignments during the labs.

5.1 Multiple incoming connections
If you use the SMTP server, there is no waiting socket, so someone else can not use the server before you close your connection. To solve this problem, you can create a separate thread for each incoming connection.

5.2 Extended SMTP features
There is much more functionality in the SMTP protocol. Choose one or more features and discuss with me if you should implement this, and how many points you can get for this.

5.3 Your own feature
Think of a feature, discuss this with me if this is doable and how many points you can get for this.

6 Assessment
The assessment of this assignment is as follows:
As you see, you can get a total of 9 point for this assignment. To get 10 points, you have to do some extra work from the bonus assignments.

You always get a 1 if you submit anything. Submitting nothing or submitting after the deadline results in a total of 0 points.

Here are the points you can earn for that:

<table>
<thead>
<tr>
<th>Part</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple connections</td>
<td>1</td>
</tr>
<tr>
<td>Implementation of SMTP extensions</td>
<td>0-1</td>
</tr>
<tr>
<td>Your own feature</td>
<td>0-1</td>
</tr>
<tr>
<td>Total:</td>
<td>1-3</td>
</tr>
</tbody>
</table>

However you can get more than 10 points, the maximum grade (of course) is a 10. You may implement multiple bonus features to make up for some lost points in other parts.

The grade will be available as soon as possible after the deadline on BlackBoard.

7 Contact

If you have any questions, feel free to ask them during the lab session. I am there to help you. If you have additional questions after the labs, you may send me an email.

Good luck with the assignment.