SMTP AND SOCKET PROGRAMMING

GP2.4 SMTP and socket programming

The goal here is twofold: on one hand, an introduction to socket programming in Python Java and, on the other hand, obtain a better understanding of SMTP protocol. This task serves as a PS and will be validated when the test mail is sent correctly.

ÁREA DE INGENIERÍA TELEMÁTICA

SMTP and socket programming

NETWORK ARCHITECTURE I

Objective

In this lab, you will be use a PC with Linux and Python interpreter. The lab goal is to send a mail by programing a TCP client in Python language. This Java programa consists of a small TCP client and a very simple command-line UI. Although no deep knowledge of Java is required, we recommend checking a good reference in Java programming like [1].

Send and email using SMTP

The next listing shows a message exchange between a SMTP client and a SMTP server. Those lines in bold correspond to messages sent by the server and those in italics to the ones the client sent. Your program must be able to reproduce this exchange, that is, at least has to understand these messages.

220 mx.uah.es Microsoft ESMTP MAIL Service, Version: 6.0.3790.4675 ready at Tue, 9 Nov 2010 09:50:31 +0100

HELO aut.uah.es

250 mx.uah.es Hello [172.29.16.42]

MAIL FROM: arqredes@aut.uah.es

250 2.1.0 arqredes@aut.uah.es....Sender OK

RCPT TO: correo_profesor@aut.uah.es

250 2.1.5 correo_profesor@aut.uah.es

DATA

354 Start mail input; end with <CRLF>.<CRLF>

See you soon

Nice to meet you

.

250 2.6.0 <X-EXF-01vJdgko3YqH0000035fa@mx.uah.es> Queued mail for delivery

QUIT

221 2.0.0 mx.uah.es Service closing transmission channel

Requirements

You have to develop the TCP client that will connect to TCP port 25 of SMTP server “correo.aut.uah.es”. If this operation succeeds, the session is opened with a greeting by the server, usually containing its fully qualified domain name (FQDN), in this case *mx.uah.es:*

220 mx.uah.es Microsoft ESMTP MAIL Service, Version: 6.0.3790.4675 ready at Tue, 9 Nov 2010 09:50:31 +0100

In this response line, first we find the status code (220), which indicates that the connection has succeeded. The rest of this response line is a textual explanation of this status code. Every SMTP response follows the same syntax: first, a 3-digit status code and second a textual explanation of the code. In this lab, you are not asked to process these messages: it is enough if you show them on screen exactly as they are received.

After the connection establishment and session initiation, your program must reproduce the session described in the previous listing (that is, HELO, MAIL FROM, RCPT TO, DATA and QUIT) and wait for the server response in every case. Opposite to the FTP session of GP7, where you introduced every message, here no user intervention is required; simply put, the program has to send those messages in the defined order and wait for the responses. This is not an interactive program; no one has to type in this commands. You can see this like a batch execution of a SMTP session (well, sort of). Finally, you have to check that the program has finally been able to send the electronic mail and that the mail has actually been received.

Starting point

We suggest to use as starting point the code at `clienteSMTP.py` shown below.

Skeleton of the solution at file named `clienteSMTP.py`.

This skeleton shows how to create the socket, how to connect to the server and how to send the first command and how to receive the first response:

```
# -*- coding: utf-8 -*-
from socket import *
msg = "\r\n Write here the content of your electronic message"
#Mail messages must end with a line containing only the character '.'
endLine = "\r\n.\r\n"
#Mail server we are going to connect to
serverName ="correo.aut.uah.es"
serverPort = 25

#We create the socket and connect to the server
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))

#220 mail10.aut.uah.es ESMTP
recv1 = clientSocket.recv(1024)
print recv1

#We first send 'HELO' and show the response
heloCommand = 'HELO aut.uah.es\r\n'

clientSocket.send(heloCommand)
recv1 = clientSocket.recv(1024)
#250 mail10.aut.uah.es
print recv1

if recv1[:3] != '250':
       print 'Expected response code not received'
       exit
```

// TO BE COMPLETED BY THE STUDENT

```
        }
}
```

Notes:

- To write a message in a socket, use the method send() as follows: clientSocket.sen`('text to be sent')`.
- To read a full line from the socket (like a response, for instance) you can use the method *recv()* as follows: clientSocket.recv`(), indicating between the parentheses the maximum number of bytes expected`.
- To tell the server that a command has finished, you have to include characters CRLF (\r\n) (i.e. carriage return and line feed) at the end of the line.
- Do not forget to close the socket at the end of the program by calling the method `.close()` of the socket.
- If you want to print something out so that you know what is going on in the program (like `printf` in C), you can use the method *print*.

Executing the program

After completing the code, you can test the program. Python is an interpreted language, then compilation is not needed. Execute the program as follows, open a terminal and write:

>Python clienteSMTP.py

The output of the program should be as follows:

```
220 mail10.correo.aut.uah.es

250 aut.uah.es

250 2.1.0 Ok

250 2.1.5 Ok

354 End data with <CR><LF>.<CR><LF>

250 2.0.0 Ok: queued as 67C6B6A073
```

References

[1] Guido van Rossum, "El tutorial de Python" (2009). Disponible en: http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf.

[2] Isabel Gracia y Andrés Marzal, 'Introducción a la Programación con Python', Publicaciones de la Universitat Jaume I. Disponible en: http://www.uji.es/bin/publ/edicions/ippython.pdf