

Introduction to SMTP and e-mail

RES, Lecture 4

Olivier Liechti



HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

Recap:

What did we put in practice
in the roulette lab??

What is an Application-Level Protocol?

- A **set of rules** that specify how the application components (e.g. clients and servers) **communicate with each other**. Typically, a protocol defines at least:
 - **Which transport-layer protocol** is used to exchange application-level messages. (e.g. TCP for HTTP)
 - **Which port number(s)** to use (e.g. 80 for HTTP)
 - **What kind of messages** are exchanged by the application components and the **structure** of these messages.
 - The **actions** that need to be taken when these messages are received and the **effect** that is expected.
 - Whether the protocol is **stateful** or **stateless**. In other words, whether the protocol requires the server to manage a session for every connected client.



blocking IO (synchronous)

n employee = n threads

employees are expensive

limited space for employees in the truck

few employees => long queue



non-blocking IO (asynchronous)

there is only 1 employee (1 thread)

customers are called back when the request is fulfilled
no queue

Multi Threaded Single Process Blocking

The 'old' Java way

The server uses a first thread to wait for connection requests from clients.

Each time a client arrives, a new thread is created and used to serve the client.

Millions of clients, millions of threads?

Resource hungry.
Not scalable.

The ReceptionistWorker implements a run() method that will execute on its own thread.

```
private class ReceptionistWorker implements Runnable {  
  
    @Override  
    public void run() {  
        ServerSocket serverSocket;  
  
        try {  
            serverSocket = new ServerSocket(port);  
        } catch (IOException ex) {  
            LOG.log(Level.SEVERE, null, ex);  
            return;  
        }  
  
        while (true) {  
            LOG.log(Level.INFO, "Waiting for a new client");  
            try {  
                Socket clientSocket = serverSocket.accept();  
                LOG.info("A new client has arrived...");  
                new Thread(new ServantWorker(clientSocket)).start();  
            } catch (IOException ex) {  
                LOG.log(Level.SEVERE, ex.getMessage(), ex);  
            }  
        }  
    }  
}
```

As soon as a client is connected, a new thread is created.
The code that manages the interaction with the client executes on this thread.

2 types of workers, n+1 threads

Single Thread Single Process Asynchronous Programming

The 'à la Node.js' way

The server uses a single thread, but in a non-blocking, asynchronous way.

Callback functions have to be written, so that they can be invoked when clients arrive, when data is received, etc.

Different programming logic.
Scalable.

We are registering callback functions on the various types of events that can be notified by the server...

```
// let's create a TCP server
var server = net.createServer();

// it reacts to events: 'listening', 'connection', 'close', etc.
// register callback functions, to be invoked when the events
// occur (everything happens on the same thread)

server.on('listening', callbackFunctionToCallWhenSocketIsBound);
server.on('connection',
  callbackFunctionToCallWhenNewClientHasArrived);

//Start listening on port 9907
server.listen(9907);

// This callback is called when the socket is bound and is in
// listening mode. We don't need to do anything special.
function callbackFunctionToCallWhenSocketIsBound() {
  console.log("The socket is bound and listening");
  console.log("Socket value: %j", server.address());
}

// This callback is called after a client has connected.
function callbackFunctionToCallWhenNewClientHasArrived(socket) {
  ...
}
```

... and we code these functions, implementing the behavior that is expected when the events occur.

This is a non-blocking call...

So, we need to pass a callback that will be called when connection is established

```
function talkToServer(data) {  
    client.connect(PORT, HOST, function() {  
        console.log('CONNECTED TO: ' + HOST + ':' + PORT);  
        //client.setNoDelay();  
        client.write('INFO\n');  
        client.write('LOAD\n');  
        client.write(data);  
        client.write('\n');  
        client.write('ENDOFDATA\n');  
        client.write('INFO\n');  
        client.write('BYE\n');  
    });  
  
    // The following code registers a callback function on an event notified by the client.  
    // When data is available for the client (i.e. bytes have arrived through the socket),  
    // the we print it on the console.  
  
    client.on('data', function(data) {  
        console.log("">>>> " + data);  
    });  
  
    client.on('end', function() {  
        console.log("** socket closed");  
    });  
}
```

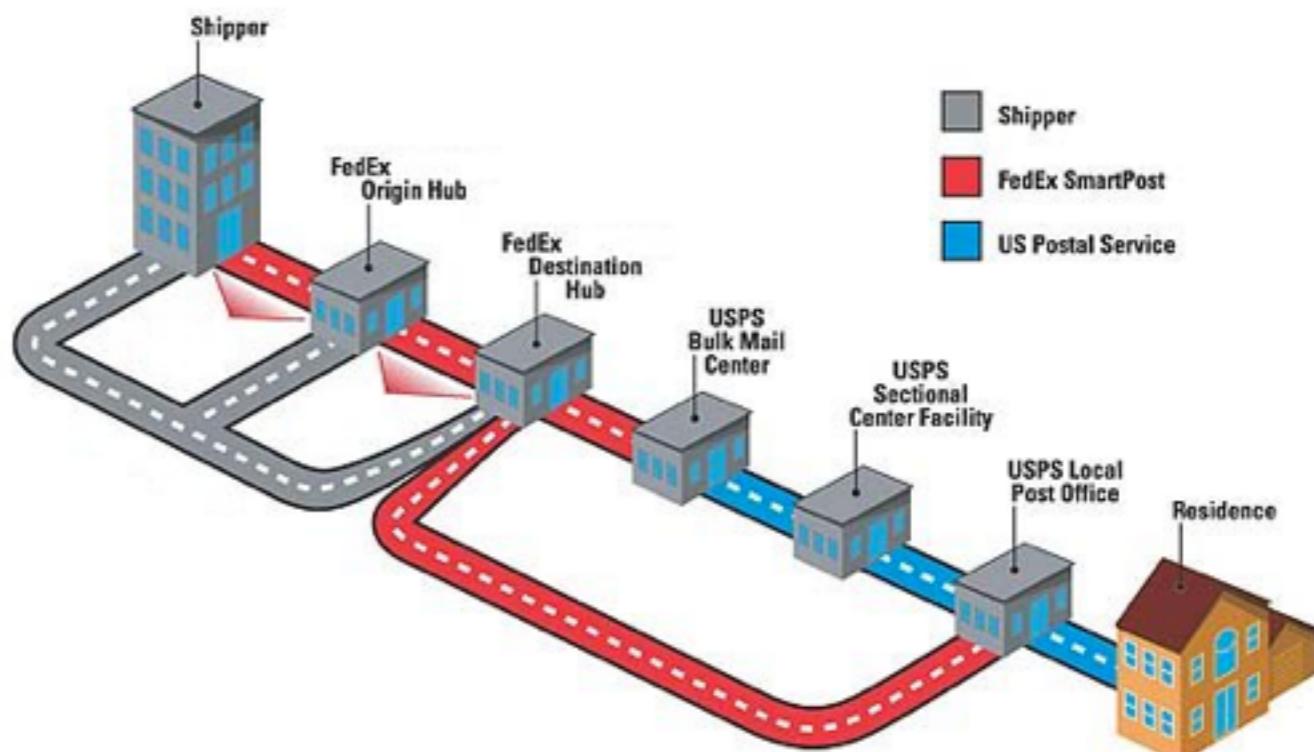
non-blocking I/Os are event-based

What else?

- What we built is a **library**, which could be used from the user interface layer (whether a GUI, a CLI, etc.).
- Using TDD to specify and build the library was helpful to **avoid coupling** between the UI and the network layers (compare with a quick-hack where the network code would be in the event handlers of buttons).
- We have seen the importance of **stable interfaces** (and seen that changing published interfaces can negatively impact a large number of people). Creating an interface is easy. Managing its lifecycle is hard.
- We have seen that unit tests can be used to specify the intended behaviour, but that this can be a bit challenging (and require more than a single test). **Having more tests reduces ambiguity.**
- **Is it a bad idea to count the number of issued commands on the client side**, if the server keeps track of them? (building the right product + building the product right).



HEIG-VD
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



Warning 1

The slides and the webcasts contain examples and demos with **real SMTP servers.**

The behaviour of these servers may change over time. It may also change depending on the network you are connected to (internal, ISP, other ISP).

The main reason why a server might behave differently is the fight between mail administrators and **spammers**.

Warning 2

It is a good thing to experiment with real SMTP servers.

But remember that they are real servers and act responsibly.

Please avoid launching a **surprise denial of service attack** with your accidental infinite loop.

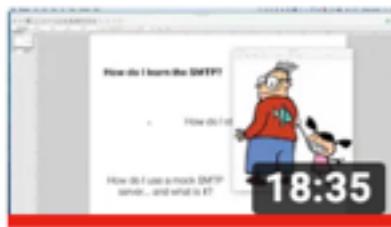
13



Labo SMTP, part 1

Olivier Liechti

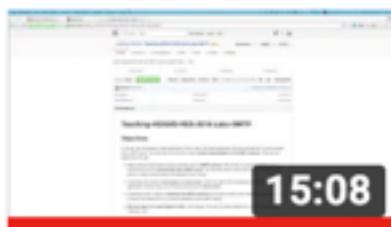
14



Labo SMTP, part 2

Olivier Liechti

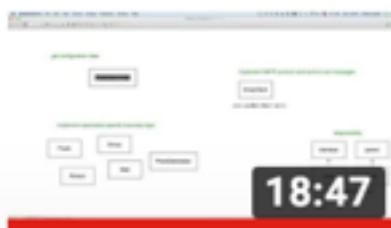
15



Labo SMTP, part 3

Olivier Liechti

16



Labo SMTP, part 4

Olivier Liechti

- SMTP demo & hints
- SMTP protocol
- Mock server
- Implementation walk-through

Tuesday	Wednesday	Friday
10.04 SMTP	11.04 M'sieur, j'ai loupé mon train. See you next year.	13.04 M'sieur, docker marche pas. See you next year.
17.04 Labo SMTP	18.04 Labo SMTP Démos “option rapide”	20.04 Labo SMTP Démos “option rapide”
24.04 Travail écrit Tout, y compris SMTP	25.04 Labo SMTP Démos “option standard”	27.04 Labo SMTP Démos “option standard”

Démo (5 minutes MAX)	
Le labo est terminé et la démo est faite le 18 ou le 24 avril.	1 pt
Le groupe arrive à démarrer un serveur mock dans un container Docker et à expliquer à quoi il sert.	1 pt
Le groupe montre comment configurer la campagne de “pranks” et lance son programme. Le groupe explique les résultats dans le serveur mock. La démo ne marche pas: 0 pt!	1.5 pt
Le groupe montre son repo GitHub. En regardant les commits, on voit que tout le monde a participé et qu'il n'y a pas seulement un gros commit à la fin.	0.5 pt
Une documentation de qualité et conforme aux exigences est fournie dans le repo GitHub.	1 pt

[https://docs.google.com/spreadsheets/d/
1KuuhFDZ85CEbqpb51mL8zuOnvbP65Nd0q
CEmYHNAQc/edit?usp=sharing](https://docs.google.com/spreadsheets/d/1KuuhFDZ85CEbqpb51mL8zuOnvbP65Nd0qCEmYHNAQc/edit?usp=sharing)



What happens when Bob wants
to **send an e-mail** to Alice?



Bob uses **Thunderbird** to write his mail.



Alice uses **MS Outlook** to check and read her mails.



In the technical specs (RFCs), these programs are called **Mail User Agents (MUA)**

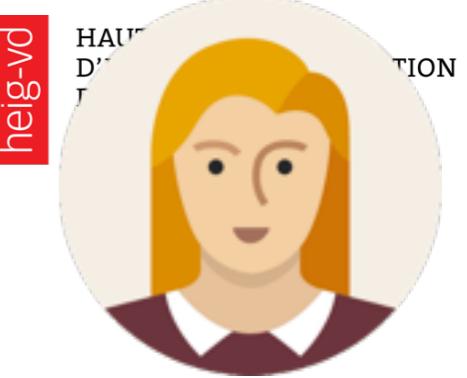


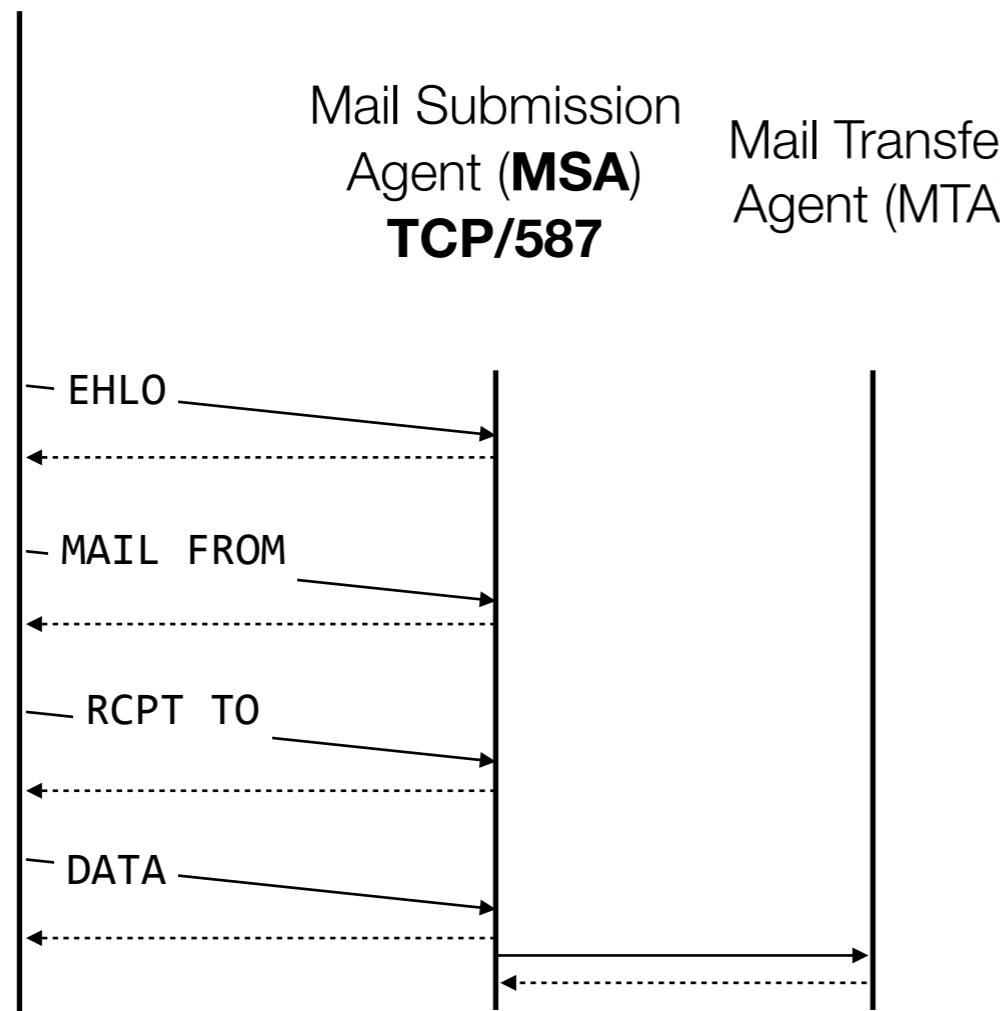


Bob uses his professional e-mail address. His company runs a **MS Exchange Server**.



Alice uses her private address. She has an account (and a **mailbox**) on the **Google gmail** infrastructure.



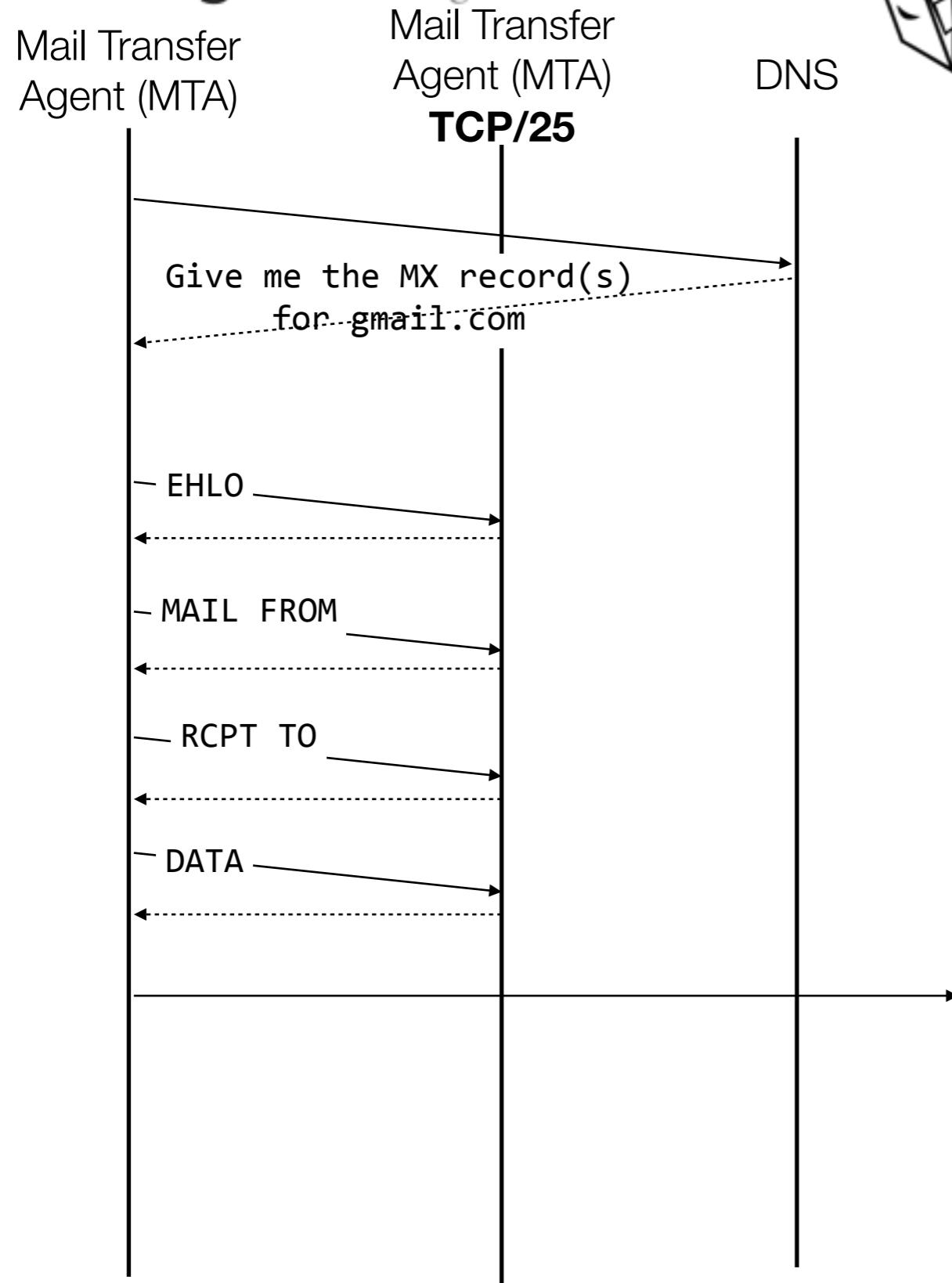


Bob writes a message to “**alice.res@gmail.com**”. He pushes on the “Send” button.

The Exchange Server is made of **2 logical components**: the **MSA** and the **MTA**.

Bob’s MUA asks Bob’s MSA to deliver the mail. It uses the **SMTP** protocol for that purpose and (should) use TCP port 587.

After enforcing **usage policies**, the MSA delegates the work to the MTA. We don’t know how.



Bob's MTA initially does not know where to forward the mail...

It issues a **DNS** query to get a list of **MX records** for Alice's domain (gmail.com).

When Bob's MTA knows the IP address of Alice's MTA, it uses the **SMTP** protocol once more to forward the message. TCP **port 25** is used in this case.

When Alice's MTA receives the mail, it stores it in Alice's **mailbox** (for later retrieval).

dig

```
DIG(1)                                admin — less + man dig — 120x30
                                         BIND9                               DIG(1)

NAME
    dig - DNS lookup utility

SYNOPSIS
    dig [@server] [-b address] [-c class] [-f filename] [-k filename] [-m] [-p port#] [-q name] [-t type]
        [-x addr] [-y [hmac:]name:key] [-4] [-6] [name] [type] [class] [queryopt...]

    dig [-h]

    dig [global-queryopt...] [query...]

DESCRIPTION
    dig (domain information groper) is a flexible tool for interrogating DNS name servers. It performs
    DNS lookups and displays the answers that are returned from the name server(s) that were queried.
    Most DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use
    and clarity of output. Other lookup tools tend to have less functionality than dig.

    Although dig is normally used with command-line arguments, it also has a batch mode of operation for
    reading lookup requests from a file. A brief summary of its command-line arguments and options is
    printed when the -h option is given. Unlike earlier versions, the BIND 9 implementation of dig allows
    multiple lookups to be issued from the command line.

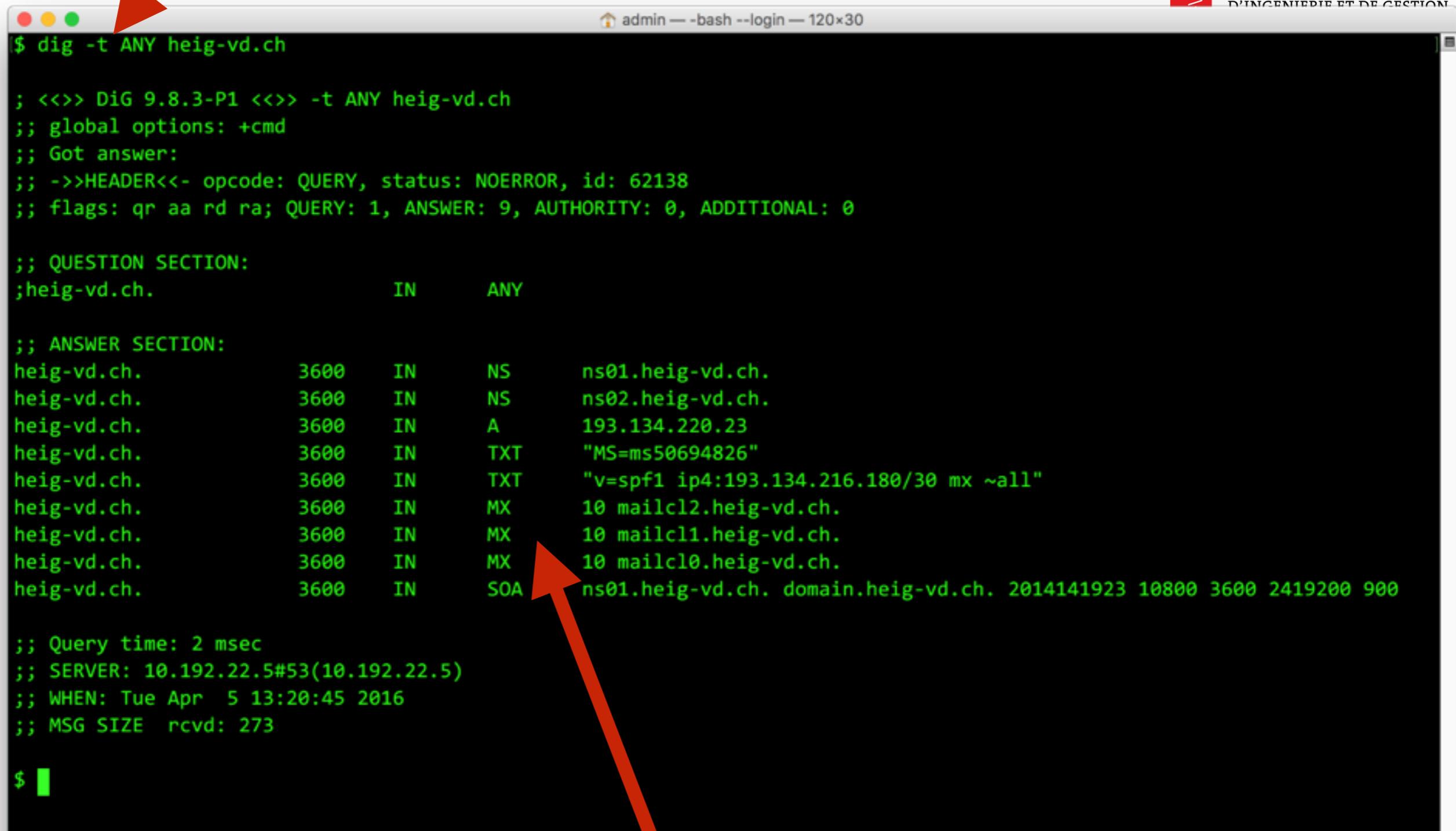
    Unless it is told to query a specific name server, dig will try each of the servers listed in
    /etc/resolv.conf.

    When no command line arguments or options are given, dig will perform an NS query for "." (the root).

:
```

nslookup is another command for querying DNS

dig -t ANY heig-vd.ch



```
$ dig -t ANY heig-vd.ch

; <>> DiG 9.8.3-P1 <>> -t ANY heig-vd.ch
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 62138
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 9, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;heig-vd.ch.           IN      ANY

;; ANSWER SECTION:
heig-vd.ch.          3600    IN      NS      ns01.heig-vd.ch.
heig-vd.ch.          3600    IN      NS      ns02.heig-vd.ch.
heig-vd.ch.          3600    IN      A       193.134.220.23
heig-vd.ch.          3600    IN      TXT     "MS=ms50694826"
heig-vd.ch.          3600    IN      TXT     "v=spf1 ip4:193.134.216.180/30 mx ~all"
heig-vd.ch.          3600    IN      MX      10 mailcl2.heig-vd.ch.
heig-vd.ch.          3600    IN      MX      10 mailcl1.heig-vd.ch.
heig-vd.ch.          3600    IN      MX      10 mailcl0.heig-vd.ch.
heig-vd.ch.          3600    IN      SOA     ns01.heig-vd.ch. domain.heig-vd.ch. 2014141923 10800 3600 2419200 900

;; Query time: 2 msec
;; SERVER: 10.192.22.5#53(10.192.22.5)
;; WHEN: Tue Apr  5 13:20:45 2016
;; MSG SIZE  rcvd: 273

$
```

MX records point to the SMTP servers for the domain

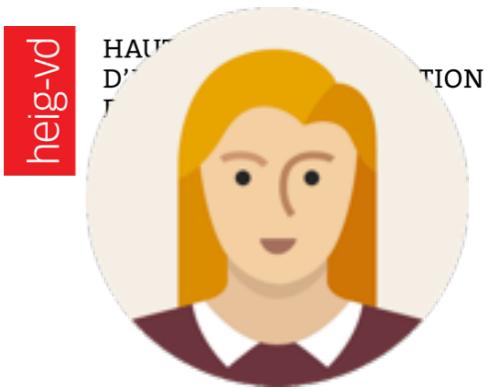


SMTP
587

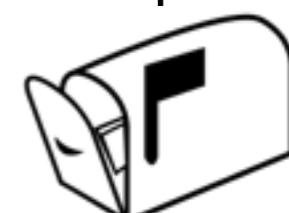


In the last step, Alice's MUA uses another protocol (e.g. IMAP, POP3) to fetch mails from the mailbox.

SMTP
25



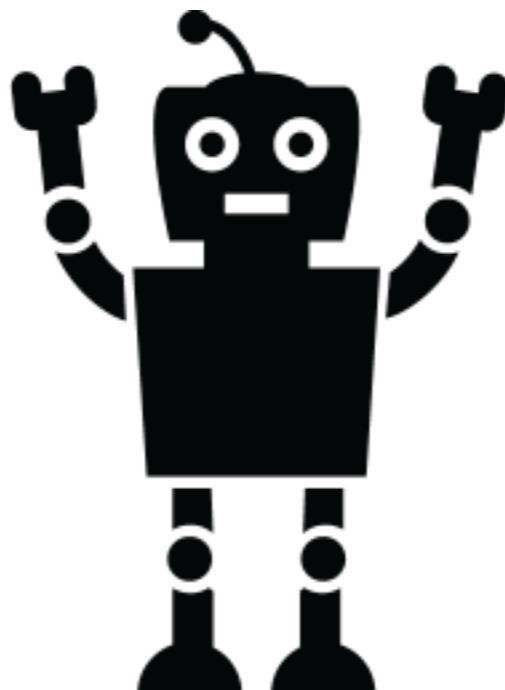
IMAP/POP3





Let's be human Exchange Servers
(and play the role of Bob's MTA).

But instead of forwarding the mail
to gmail, let's forward the mail via
the **HEIG-VD's SMTP** server.



```
dig -t MX heig-vd.ch
```

```
heig-vd.ch. 3600 INMX 10 mailcl0.heig-vd.ch.
```

```
telnet mailcl0.heig-vd.ch 25
```

```
openssl s_client -starttls smtp -crlf -connect  
mailcl0.heig-vd.ch:25
```

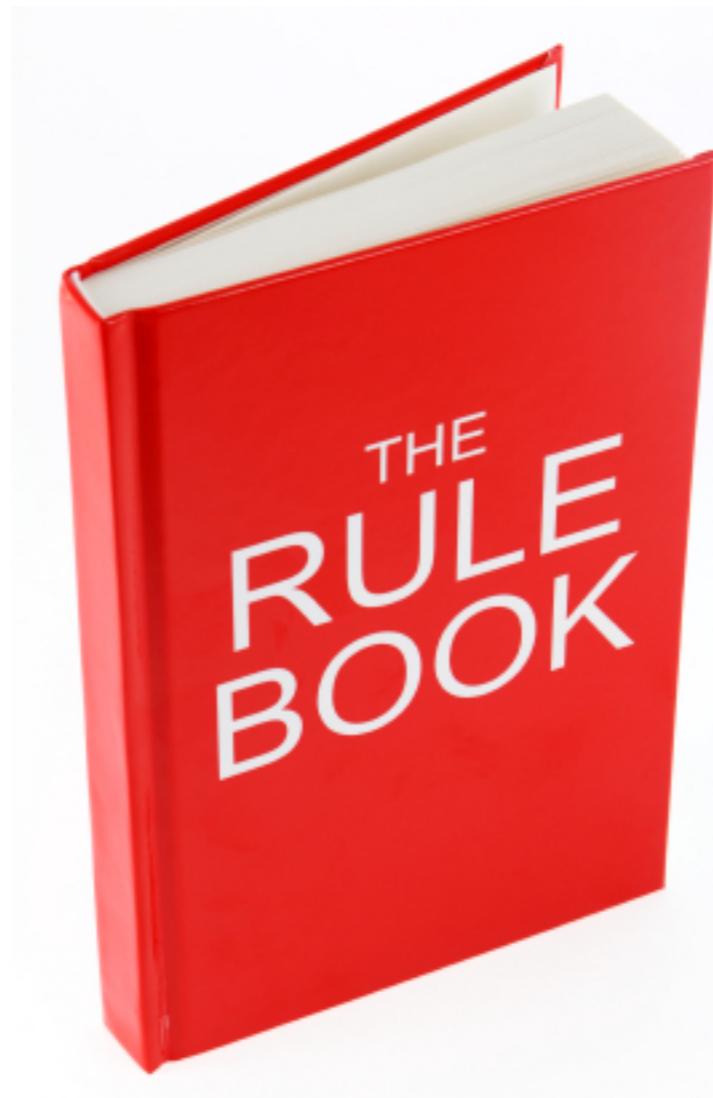
```
EHLO mycompany.com
```

```
$ telnet mailcl10.heig-vd.ch 25
mailcl10.heig-vd.ch: nodename nor servname provided, or not known
$ telnet mailcl0.heig-vd.ch 25
Trying 193.134.216.181...
Connected to mailcl0.heig-vd.ch.
Escape character is '^]'.
220 heig-vd.ch ESMTP MailCleaner (Enterprise Edition 2016.01) Tue, 05 Apr 2016 14:18:24
+0200
EHLO mycompany.com
250-heig-vd.ch Hello mbp-de-admin.einet.ad.eivd.ch [10.192.116.92]
250-SIZE 20480000
250-8BITMIME
250-PIPELINING
250-AUTH PLAIN LOGIN
250-STARTTLS
250 HELP
MAIL FROM: bob@bob.com
250 OK
RCPT TO: olivier.liechti@wasabi-tech.com
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
From: bob@areyousure.com
To: olivier.liechti@wasabi-tech.com
Subject: demo

Ok. Cool. Bye.

.
250 OK id=1anPx9-0003KC-BC
quit
221 heig-vd.ch closing connection
Connection closed by foreign host.
```

SMTP command
!=
Message header



The Specs

Table of Contents

	RFC 5321	SMTP	October 2008
<u>1. Introduction</u>	<u>5</u>		
<u>1.1. Transport of Electronic Mail</u>	<u>5</u>		
<u>1.2. History and Context for This Document</u>	<u>5</u>		
<u>1.3. Document Conventions</u>	<u>6</u>		
<u>2. The SMTP Model</u>	<u>7</u>		
<u>2.1. Basic Structure</u>	<u>7</u>		
<u>2.2. The Extension Model</u>	<u>9</u>		
<u>2.2.1. Background</u>	<u>9</u>		
<u>2.2.2. Definition and Registration of Extensions</u>	<u>10</u>		
<u>2.2.3. Special Issues with Extensions</u>	<u>11</u>		
<u>2.3. SMTP Terminology</u>	<u>11</u>		
<u>2.3.1. Mail Objects</u>	<u>11</u>		
<u>2.3.2. Senders and Receivers</u>	<u>12</u>		
<u>2.3.3. Mail Agents and Message Stores</u>	<u>12</u>		
<u>2.3.4. Host</u>	<u>13</u>		
<u>2.3.5. Domain Names</u>	<u>13</u>		
<u>2.3.6. Buffer and State Table</u>	<u>14</u>		
<u>2.3.7. Commands and Replies</u>	<u>14</u>		
<u>2.3.8. Lines</u>	<u>14</u>		
<u>2.3.9. Message Content and Mail Data</u>	<u>15</u>		
<u>2.3.10. Originator, Delivery, Relay, and Gateway Systems</u>	<u>15</u>		
<u>2.3.11. Mailbox and Address</u>	<u>15</u>		
<u>2.4. General Syntax Principles and Transaction Model</u>	<u>16</u>		
<u>3. The SMTP Procedures: An Overview</u>	<u>17</u>		
<u>3.1. Session Initiation</u>	<u>18</u>		
<u>3.2. Client Initiation</u>	<u>18</u>		
<u>3.3. Mail Transactions</u>	<u>19</u>		
<u>3.4. Forwarding for Address Correction or Updating</u>	<u>21</u>		
<u>3.5. Commands for Debugging Addresses</u>	<u>22</u>		
<u>3.5.1. Overview</u>	<u>22</u>		
<u>3.5.2. VRFY Normal Response</u>	<u>24</u>		
<u>3.5.3. Meaning of VRFY or EXPN Success Response</u>	<u>25</u>		
<u>3.5.4. Semantics and Applications of EXPN</u>	<u>26</u>		
<u>3.6. Relaying and Mail Routing</u>	<u>26</u>		
<u>3.6.1. Source Routes and Relaying</u>	<u>26</u>		
<u>3.6.2. Mail eXchange Records and Relaying</u>	<u>26</u>		
<u>3.6.3. Message Submission Servers as Relays</u>	<u>27</u>		
<u>3.7. Mail Gateways</u>	<u>28</u>		
<u>3.7.1. Header Fields in Gateways</u>	<u>28</u>		
<u>3.7.2. Received Lines in Gateways</u>	<u>29</u>		
<u>3.7.3. Addresses in Gateways</u>	<u>29</u>		
<u>3.7.4. Other Header Fields in Gateways</u>	<u>29</u>		
<u>3.7.5. Envelopes in Gateways</u>	<u>30</u>		
<u>3.8. Terminating Sessions and Connections</u>	<u>30</u>		
<u>3.9. Mailing Lists and Aliases</u>	<u>31</u>		
<u>3.9.1. Alias</u>	<u>31</u>		
<u>3.9.2. List</u>	<u>31</u>		
<u>4. The SMTP Specifications</u>	<u>32</u>		
<u>4.1. SMTP Commands</u>	<u>32</u>		
<u>4.1.1. Command Semantics and Syntax</u>	<u>32</u>		
<u>4.1.2. Command Argument Syntax</u>	<u>41</u>		
<u>4.1.3. Address Literals</u>	<u>43</u>		
<u>4.1.4. Order of Commands</u>	<u>44</u>		
<u>4.1.5. Private-Use Commands</u>	<u>46</u>		
<u>4.2. SMTP Replies</u>	<u>46</u>		
<u>4.2.1. Reply Code Severities and Theory</u>	<u>48</u>		
<u>4.2.2. Reply Codes by Function Groups</u>	<u>50</u>		
<u>4.2.3. Reply Codes in Numeric Order</u>	<u>52</u>		
<u>4.2.4. Reply Code 502</u>	<u>53</u>		
<u>4.2.5. Reply Codes after DATA and the Subsequent <CRLF>.<CRLF></u>	<u>53</u>		
<u>4.3. Sequencing of Commands and Replies</u>	<u>54</u>		
<u>4.3.1. Sequencing Overview</u>	<u>54</u>		
<u>4.3.2. Command-Reply Sequences</u>	<u>55</u>		
<u>4.4. Trace Information</u>	<u>57</u>		
<u>4.5. Additional Implementation Issues</u>	<u>61</u>		
<u>4.5.1. Minimum Implementation</u>	<u>61</u>		
<u>4.5.2. Transparency</u>	<u>62</u>		
<u>4.5.3. Sizes and Timeouts</u>	<u>62</u>		
<u>4.5.3.1. Size Limits and Minimums</u>	<u>62</u>		
<u>4.5.3.1.1. Local-part</u>	<u>63</u>		
<u>4.5.3.1.2. Domain</u>	<u>63</u>		
<u>4.5.3.1.3. Path</u>	<u>63</u>		
<u>4.5.3.1.4. Command Line</u>	<u>63</u>		
<u>4.5.3.1.5. Reply Line</u>	<u>63</u>		
<u>4.5.3.1.6. Text Line</u>	<u>63</u>		
<u>4.5.3.1.7. Message Content</u>	<u>63</u>		
<u>4.5.3.1.8. Recipients Buffer</u>	<u>64</u>		
<u>4.5.3.1.9. Treatment When Limits Exceeded</u>	<u>64</u>		
<u>4.5.3.1.10. Too Many Recipients Code</u>	<u>64</u>		
<u>4.5.3.2. Timeouts</u>	<u>65</u>		
<u>4.5.3.2.1. Initial 220 Message: 5 Minutes</u>	<u>65</u>		
<u>4.5.3.2.2. MAIL Command: 5 Minutes</u>	<u>65</u>		
<u>4.5.3.2.3. RCPT Command: 5 Minutes</u>	<u>65</u>		
<u>4.5.3.2.4. DATA Initiation: 2 Minutes</u>	<u>66</u>		
<u>4.5.3.2.5. Data Block: 3 Minutes</u>	<u>66</u>		
<u>4.5.3.2.6. DATA Termination: 10 Minutes</u>	<u>66</u>		
<u>4.5.3.2.7. Server Timeout: 5 Minutes</u>	<u>66</u>		
<u>4.5.4. Retry Strategies</u>	<u>66</u>		
<u>4.5.5. Messages with a Null Reverse-Path</u>	<u>68</u>		
<u>5. Address Resolution and Mail Handling</u>	<u>69</u>		
<u>5.1. Locating the Target Host</u>	<u>69</u>		
<u>5.2. IPv6 and MX Records</u>	<u>71</u>		
<u>6. Problem Detection and Handling</u>	<u>71</u>		

D.1. A Typical SMTP Transaction Scenario

This SMTP example shows mail sent by Smith at host bar.com, and to Jones, Green, and Brown at host foo.com. Here we assume that host bar.com contacts host foo.com directly. The mail is accepted for Jones and Brown. Green does not have a mailbox at host foo.com.

```
S: 220 foo.com Simple Mail Transfer Service Ready
C: EHLO bar.com
S: 250-foo.com greets bar.com
S: 250-8BITMIME
S: 250-SIZE
S: 250-DSN
S: 250 HELP
C: MAIL FROM:<Smith@bar.com>
S: 250 OK
C: RCPT TO:<Jones@foo.com>
S: 250 OK
C: RCPT TO:<Green@foo.com>
S: 550 No such user here
C: RCPT TO:<Brown@foo.com>
S: 250 OK
C: DATA
S: 354 Start mail input; end with <CRLF>. <CRLF>
C: Blah blah blah...
C: ...etc. etc. etc.
C: .
S: 250 OK
C: QUIT
S: 221 foo.com Service closing transmission channel
```



SMTP Servers for experiments

Mon site | Mes liens | Bienvenue LIECHTI Olivier

Quicklinks
La boîte à outils des services HEIG-VD 

heig-vd HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD www.heig-vd.ch

Services Départements Ra&D Académique Campus Infrastructure Liens Conseil représentatif Rechercher 

Intranet HEIG-VD > Services > Informatique > Poste de travail > Messagerie > Configuration > **SMTP**

Webmail
Outlook
Configuration
Exchange
IMAP
POP
SMTP
LDAP
Archivage
Spam
Mailing
Réservation salles

Le protocole SMTP est utilisé lors de l'envoi des mails. Il est complémentaire aux protocoles POP et IMAP qui ne s'occupent que de la réception.

L'adresse de notre serveur SMTP est :

smtp.heig-vd.ch

ATTENTION !!

Pour le moment, la connexion IMAP impose d'activer le SLL.

Attention, pour lutter contre le spam ainsi que pour des raisons de sécurité, notre serveur smtp, comme bien d'autres, n'autorisent l'envoi d'e-mails que depuis l'intérieur de notre réseau (ou via vpn). Depuis chez vous, il faut utiliser le serveur smtp de votre fournisseur d'accès internet.

Pour en savoir plus : [wikipedia](#)



With this default setup, you will not be able to login with your user id / password.

My Account Sign-in & security

Welcome

Sign-in & security

- Signing in to Google
- Device activity & security events
- Apps with account access

Personal info & privacy

- Your personal info
- Contacts
- Manage your Google activity
- Ads Settings
- Control your content

Account preferences

- Payments
- Language & Input Tools
- Accessibility
- Your Google Drive storage

Allow less secure apps: OFF 

Some apps and devices use less secure sign-in technology, which could leave your account vulnerable. You can turn off access for these apps (which we recommend) or choose to use them despite the risks.

Check your privacy settings →

 Your security comes first in everything we do.
[LEARN MORE](#)

WORLDWIDE SMTP DELIVERY

Like a first-class courier, for email.

Easily send and track all of your emails, and forget headaches
with email delivery.

[See Plans & Pricing](#)

[Try SMTP2GO Free](#)





**"Mon métier,
c'est Johnny,"**

Portrait Johnny VEGAS

photo : nice matin



Mock Servers

[tweakers-dev / MockMock](#)

[Code](#) [Issues 2](#) [Pull requests 4](#) [Projects 0](#) [Wiki](#) [Insights](#)

[Watch ▾](#) 10 [Unstar](#) 39 [Fork](#) 24

A mock SMTP server built with Java

[MockMock](#) [Home](#) [MockMock on Github](#)

I've got 24 mails for you. Nice!

[Delete all](#)

From	To	Subject
John Doe <someone@example.org>	Some Dude <dude@examp...	Well, this is a nice subject...
John Doe <someone@example.org>	Some Dude <dude@examp...	LOL omg!
John Doe <someone@example.org>	Some Dude <dude@examp...	The iPhone 5 is huge!
John Doe <someone@example.org>	Some Dude <dude@examp...	Did you see the new MockMock version already?
John Doe <someone@example.org>	Some Dude <dude@examp...	Well, this is a nice subject...
John Doe <someone@example.org>	Some Dude <dude@examp...	Well, this is a nice subject...
John Doe <someone@example.org>	Some Dude <dude@examp...	Did you see the new MockMock version already?



The image shows the homepage of Mailtrap. At the top, there is a navigation bar with links: HOW IT WORKS, PRICING, API, BLOG, FAQ, and HELP. There are also 'Log in' and 'Sign up' buttons. The main visual features a large, friendly cartoon character with a teal head, black body, and a wide, toothy grin. The character is wearing a black beret and holding a newspaper. It is surrounded by several floating envelopes with faces, some of which appear to be crying or distressed. To the right of the character, the word 'SAFE' is written in large, bold, teal capital letters, followed by the text 'email testing for dev teams' in a smaller, italicized serif font. Below this text is a laptop screen displaying a video player interface with a play button and the text 'See How it Works'. The video player has three buttons: 'Watch' (with a play icon), 'Video' (with a play icon), and another 'Video' button. The overall theme is playful and professional, targeting developers.