EMAIL

INTRODUCTION TO MAIL TRANSFER PROTOCOLS FOR THE INTERNET

Peter R. Egli
INDIGOO.COM

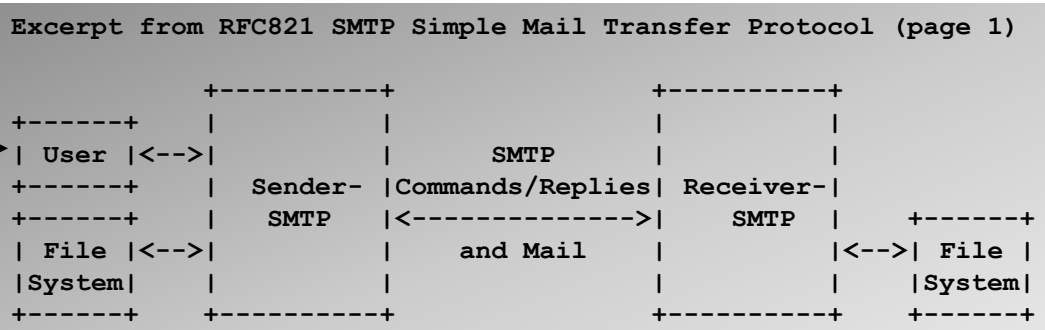
Contents

1. EMail Electronic Mail RFC821/RFC822
2. Email Elements
3. Email Message
4. Email Transfer with SMTP
5. Email Retrieval with POP3 Post Office Protocol RFC1939
6. Email Retrieval with IMAP4 Internet Mail Access Protocol RFC2060
7. POP versus IMAP
8. Email Address
9. MIME Multipurpose Internet Mail Extensions

1. Email Electronic Mail RFC821 / RFC822 (1/5)

Email the old days (back in 1982, RFC821 SMTP SIMPLE MAIL TRANSFER PROTOCOL by the venerable Jon B. Postel) was a very simple system as shown in RFC821:

Excerpt from RFC821 SMTP Simple Mail Transfer Protocol (page 1)



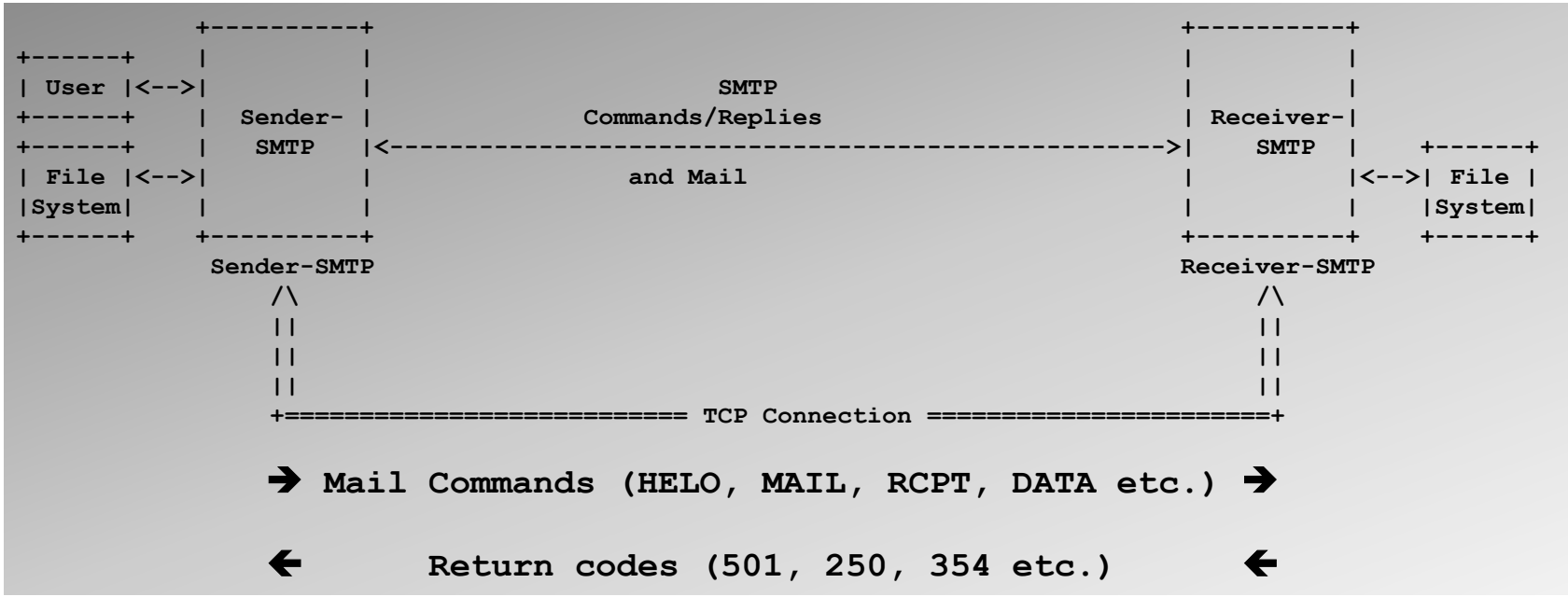
Model for SMTP Use
Figure 1

- **The email client sits on same machine as email server (sender SMTP).**
- **The emails are sent to a receiver-SMTP and stored in the file system.**
- **With SMTP, mail was directly transferred from the sending user‘s host to the receiving user‘s host.**
- **RFC821 SMTP: defines commands and procedures for the transfer of electronic messages (envelope). Obsoleted by RFC2821.**
- **RFC822: defines the format of electronic messages (header and body). RFC822 was obsoleted by RFC2822.**

1. Email Electronic Mail RFC821 / RFC822 (2/5)

RFC821 (Simple Mail Transfer Protocol) defines the commands response codes between SMTP sender and receiver.

ASCII commands are sent from SMTP sender to SMTP receiver.
The SMPT receiver evaluates the commands and sends back return codes.



1. Email Electronic Mail RFC821 / RFC822 (3/5)

RFC821 SMTP mandatory commands:

HELO
Hello. Used to identify the sender to the receiver. This command must accompany the hostname of the sending host. In the extended protocol (ESTMP), the command EHLO is used instead. See the "Extended SMTP" section later in the chapter for more information.

MAIL
Initiates a mail transaction. Arguments include the "from" field or the sender of the mail.

RCPT
Identifies the recipient of the message.

DATA
Announces the beginning of the actual mail data (header and body of the message). The data can contain any 128-bit ASCII code and is terminated with a single line containing a period (.).

RSET
Aborts (resets) the current transaction.

VRFY
Used to confirm a recipient user.

NOOP
This "no operation" command specifies no action.

QUIT
Closes the connection.

SEND
Lets the receiving host know that the message must be sent to another terminal.

1. Email Electronic Mail RFC821 / RFC822 (4/5)

RFC821 SMTP optional extended commands:

The following commands are specified, but not required, by RFC 821:

SOML
Send or mail. Tells the receiving host that the message must be sent to other terminals or mailboxes.

SAML
Send and mail. Tells the receiving host that the message must be sent to other terminals and mailboxes.

EXPN
Used to expand a mailing list.

HELP
Requests helpful information from the receiving host.

TURN
Requests that the receiving host take on the role of the sending host.

1. Email Electronic Mail RFC821 / RFC822 (5/5)

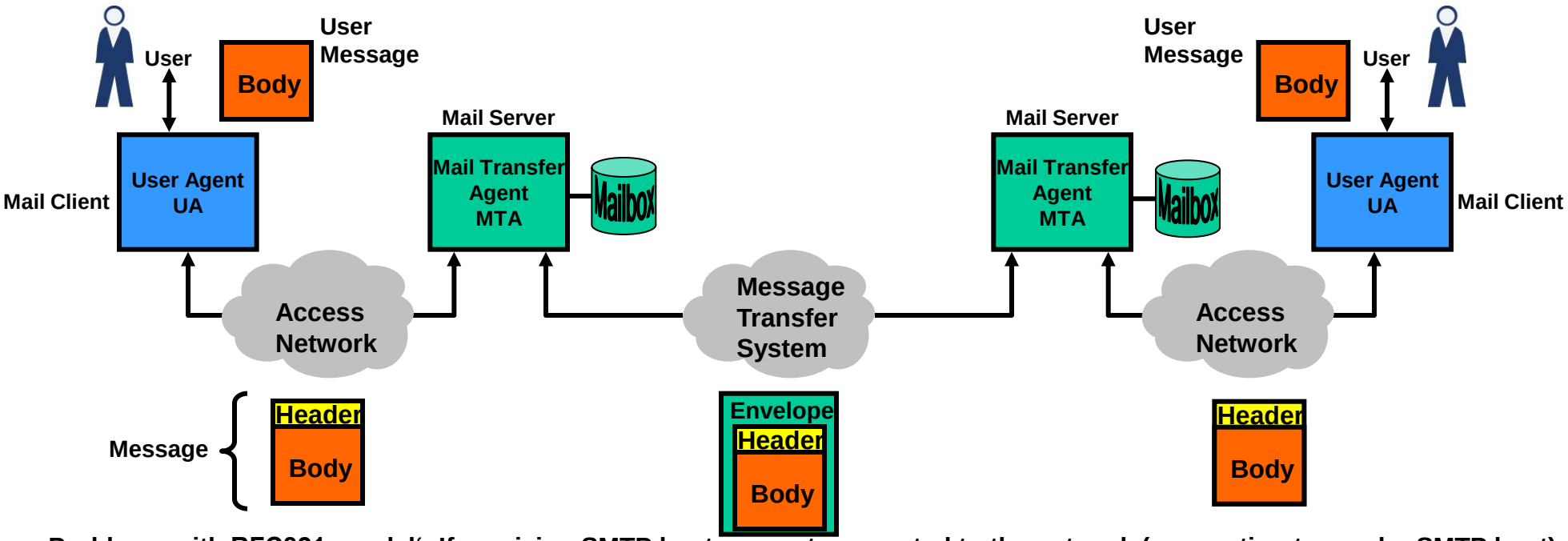
RFC821 SMTP return codes:

Code	Description
211	Help reply, system status
214	Help message
220	Service ready
221	Closing connection
250	Requested action okay
251	User not local, forwarding to
354	Start mail input
421	Service not available
450	Action not taken, mailbox busy
451	Action aborted, local error
452	Action not taken, insufficient storage
500	Command unrecognized or syntax error
501	Syntax error in parameters or arguments
502	Command not supported
503	Bad sequence of commands (given out of order)
504	Command parameter not supported
550	Action not taken, mailbox unavailable
551	Not a local user
552	Aborted: Exceeded storage allocation
553	Action not taken, mailbox name not allowed
554	Transaction failed

Code groups:
5xx for failure,
4xx for temporary problem,
1xx-3xx for success

2. Email Elements (1/3)

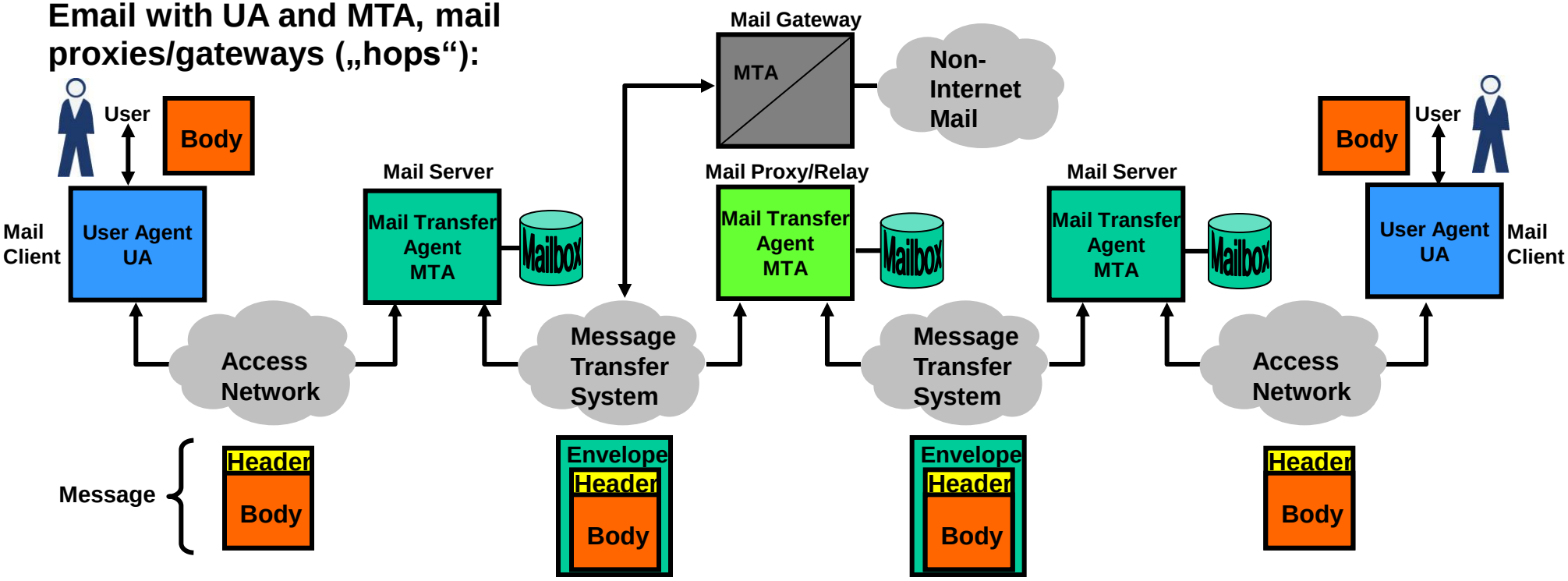
Email with User Agent UA and Mail Transfer Agent MTA:



- **Problems with RFC821 ‚model': If receiving SMTP host was not connected to the network (connection to sender SMTP host) mail could not be transferred.**
- **Solution: Mail Transfer Agent MTA (=mail server) tries to send mail on behalf of User Agent (=mail client).**
- **MTA repeatedly tries to send the email until successfully delivered or until timeout occurred.**
- **User Agent UA needs to be online only for retrieving / sending mail.**
- **The protocol between UA and MTA is either SMTP/POP/IMAP or a vendor specific proprietary mail transfer protocol.**

2. Email Elements (2/3)

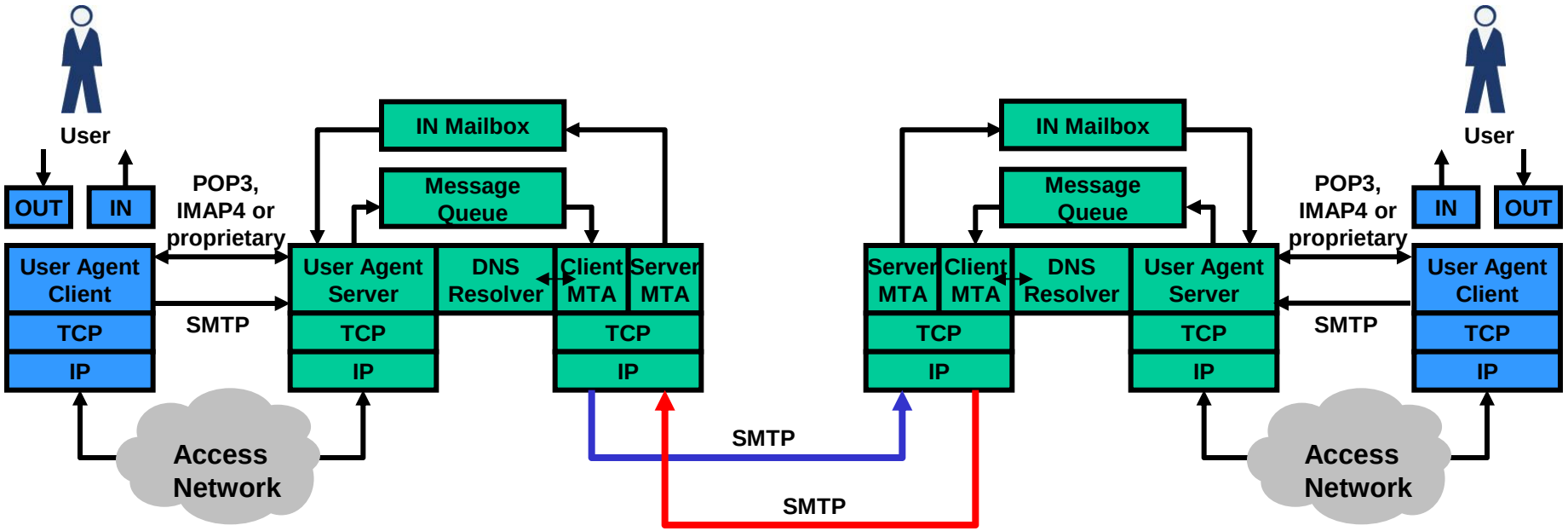
Email with UA and MTA, mail proxies/gateways („hops“):



- Any number of mail proxies (relays, hops) can be in the path from sender to receiver of email:
  E.g. mail proxy as mail relay between Internet and mail server in LAN.
- Mail server (mail host, mail gateway) inspects “RCPT TO:” address (someone@somewhere.com). If the addressee is on the local server the mail is delivered into the user’s inbox. If the addressee is not on the local server the mail is forwarded to the next mail server (if the server is configured to do so).
- A mail proxy/relay is simply an SMTP receiver/sender that is enabled to forward mail messages.
- A mail gateway converts mail between different formats/protocols.

2. Email Elements (3/3)

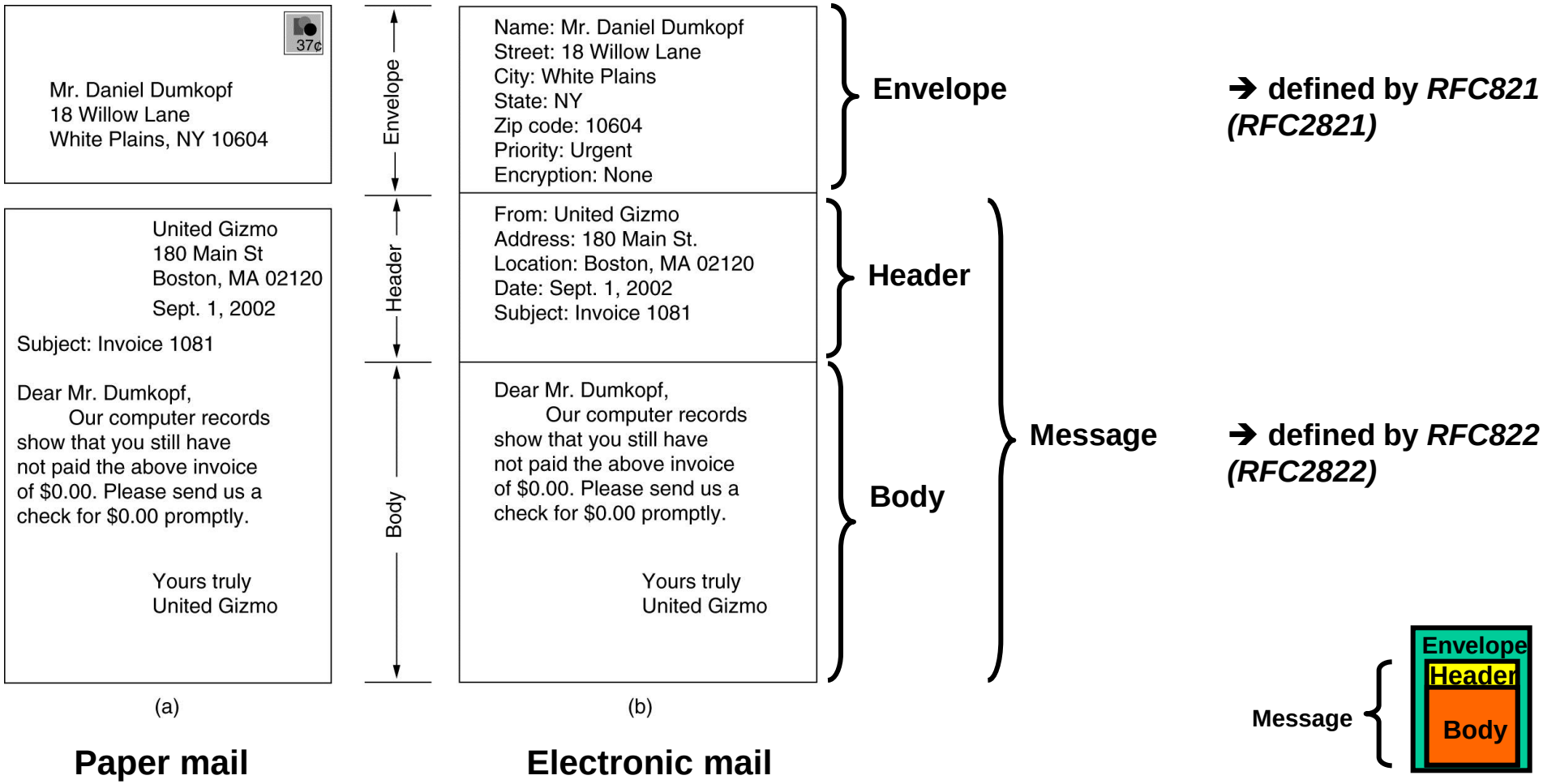
Email elements and protocol stacks in more detail.



- **User Agent Client: communicates with mail server through SMTP (send mail) or POP/IMAP (mail retrieval).**
- **User Agent Server: receives emails from user (through SMTP or proprietary protocol) and delivers mails to user via POP or IMAP.**
- **Message Queue: „outbox“ of unsent emails that await delivery to receiving mail server.**
- **Client MTA: periodically tries to send email messages in message queue until successful or until messages time out.**
- **Server MTA: accepts incoming SMTP connection requests and stores incoming emails in IN Mailbox (inbox).**
- **Resolver: DNS resolver that performs a look up from domain name to IP address.**

3. Email Message (1/3)

Envelope, message header and message body, a letter (a) versus a mail message (b):



3. Email Message (2/3)

RFC822 (RFC2822) header fields related to message transport:

Header	Meaning
To:	E-mail address(es) of primary recipient(s)
Cc:	E-mail address(es) of secondary recipient(s)
Bcc:	E-mail address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	E-mail address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

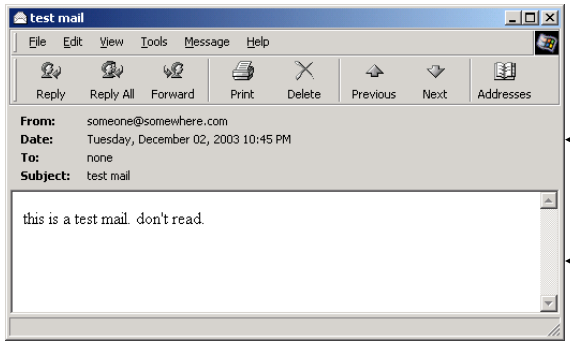
Other RFC822 (RFC2822) header fields:

Header	Meaning
Date:	The date and time the message was sent
Reply-To:	E-mail address to which replies should be sent
Message-Id:	Unique number for referencing this message later
In-Reply-To:	Message-Id of the message to which this is a reply
References:	Other relevant Message-Ids
Keywords:	User-chosen keywords
Subject:	Short summary of the message for the one-line display

3. Email Message (3/3)

RFC822 mail message format:

Header fields are of the format:
<field name>: <field value> CRLF
Entire mail:
Lines max. 1000 chars with CR.
7 Bit ASCII.
Body separated from header with an empty line (<CRLF>).
RFC821 end of mail message: empty line with dot followed by <CRLF> („.CRLF‘).



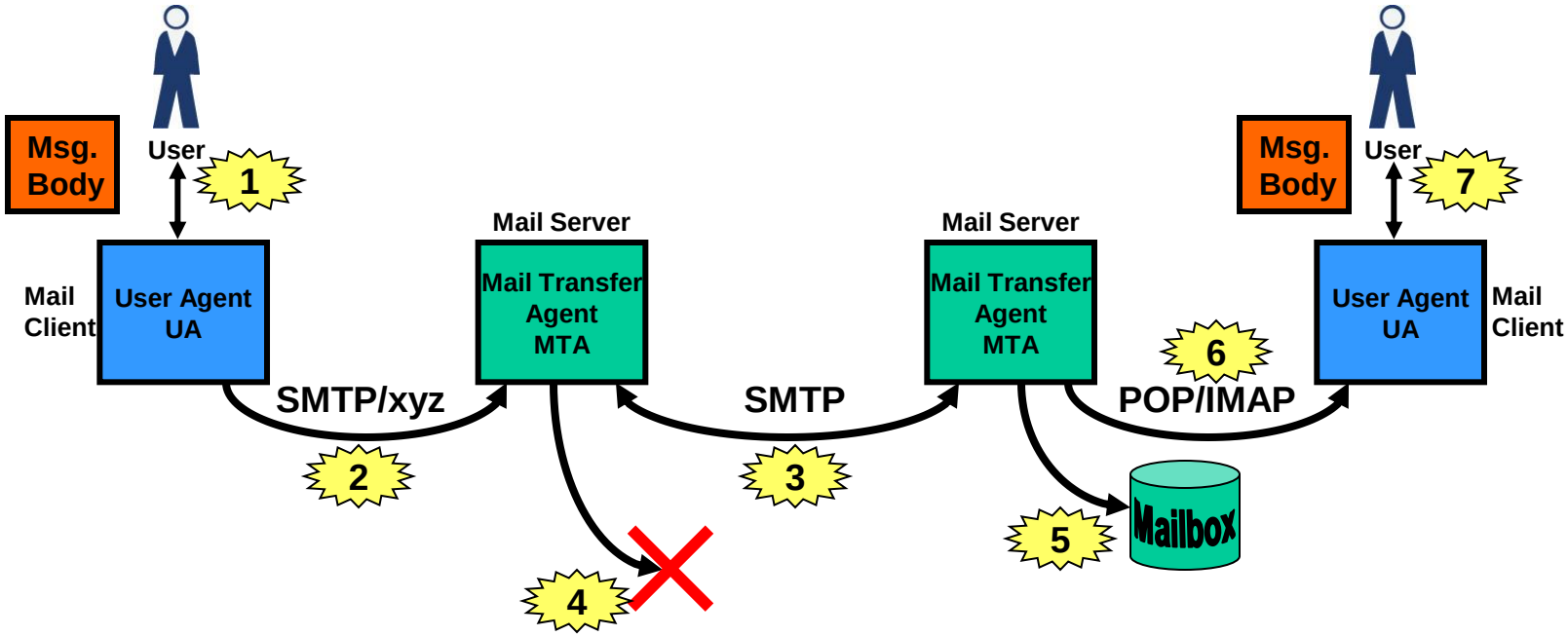

```
Received: from  ([127.0.0.1]) by mydomain with Microsoft SMTPSVC(5.0.2195.5329);
          Tue, 3 Dec 2002 08:14:35 +0100
from: someone@nowhere.com
to: anybody@nowhere.com
Return-Path: someone@nowhere.com
Message-ID: <FAREUcP6yYYg7hsdgbj00000001@mydomain>
X-OriginalArrivalTime: 04 Dec 2003 07:15:12.0737 (UTC) FILETIME=[5BC87510:01C3BA36]
Date: 3 Dec 2002 08:15:12 +0100

This is a test mail.
Don’t read it.
.
```

Header

Body

A blank line separates header and body.

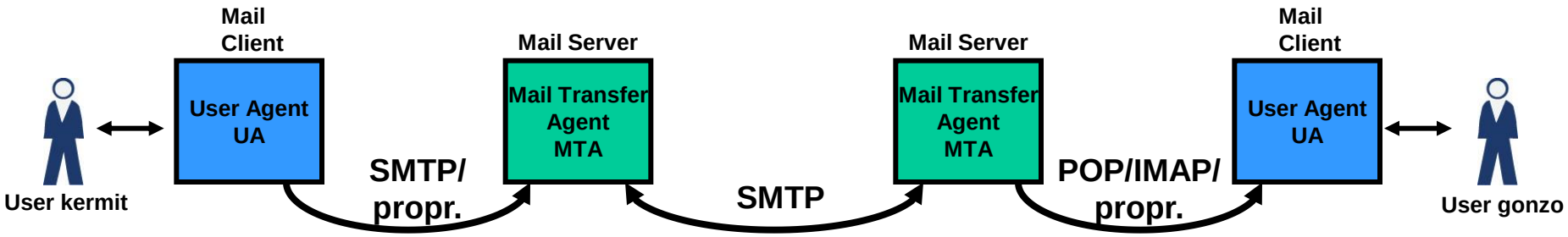
4. Email Transfer with SMTP (1/5)



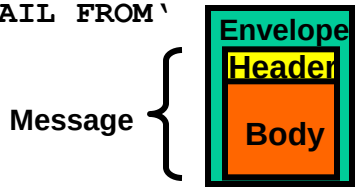
1. Email message (body) is edited by user and stored as file (with message header).
2. UA sends message (body+message) by SMTP or another protocol to mail server.
3. Mail server resolves name (user@somewhere.ch) and tries to deliver email to destination server. The message possibly takes a hop-by-hop path, travelling from mail server to mail server.
4. If the email can not be delivered the originating mail server returns an error message („the message could not be delivered for the past 4 hours...“).
5. The receiving (destination) mail server stores the message into the user‘s inbox. If the user does not exist an error message is returned.
6. When the user reads emails they are transferred by POP/IMAP from the server to the mail client.
7. The message is read by the user.

4. Email Transfer with SMTP (2/5)

Mapping of fields:

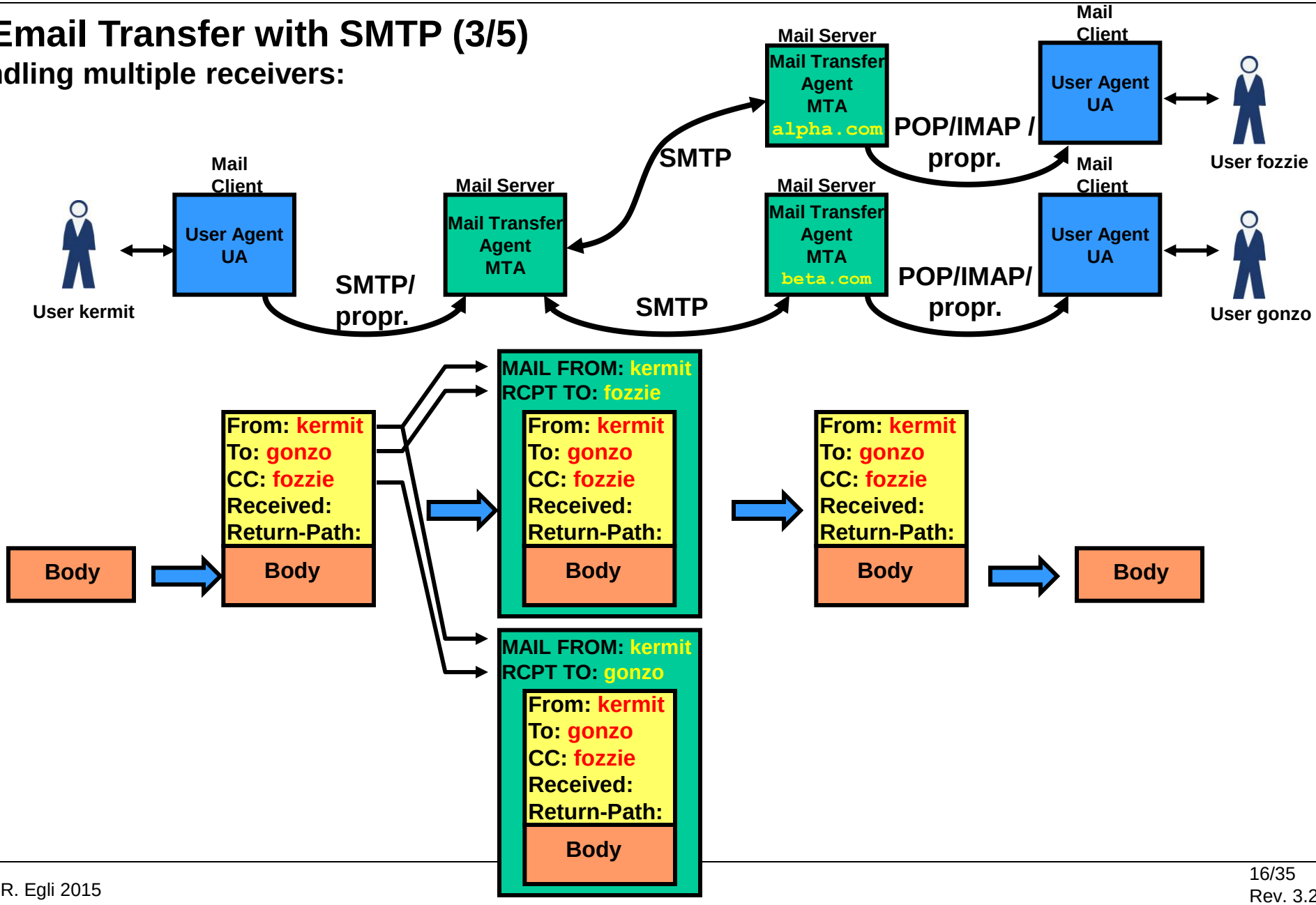


➔ Mail Transfer Agent MTA maps the `From:` and `To:` fields from the message header to the envelope ‚`MAIL FROM`' and '`RCPT TO`' fields.



4. Email Transfer with SMTP (3/5)

Handling multiple receivers:



4. Email Transfer with SMTP (4/5)

TELNET as simple mail UA (User Agent):

SMTP RFC821 uses NVT (Network Virtual Terminal). NVT is:

- **A terminal that sends/receives ASCII characters (7bit chars).**
- **End-of-line is sent as <CR><LF> (carriage return, line feed) combination.**

TELNET (RFC854) uses NVT too. Thus email can be sent with TELNET as user agent.

```
Sample SMTP session with TELNET (to incoming SMTP server of FHZH):
cmd>telnet mail.indigoo.com 25
S: 220 mail.indigoo.com ESMTP Sendmail 8.12.9+Sun/8.12.9; Tue, 2 Dec 2003 21:52:55 +0100 (CET)
C: HELO pegli
S: 250 mail.indigoo.com Hello dclient80-218-75-247.hispeed.ch [80.218.75.247], pleased to meet you
C: MAIL FROM: <someone@somewhere.com>
S: 250 2.1.0 someone@somewhere.com... Sender ok
C: RCPT TO: <pegli@fhzh.ch>
S: 250 2.1.5 pegli@fhzh.ch... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Subject: hello pegli
C:
C: good morning,
C:
C: this is a test message.
C:
C: und tschuess
C: .
S: 250 2.0.0 hB2KqtXI020827 Message accepted for delivery
```

How to find the IP address of a domain's email server:
Use nslookup, search for MX DNS record type of the domain.

4. Email Transfer with SMTP (5/5)

SMTP Summary:

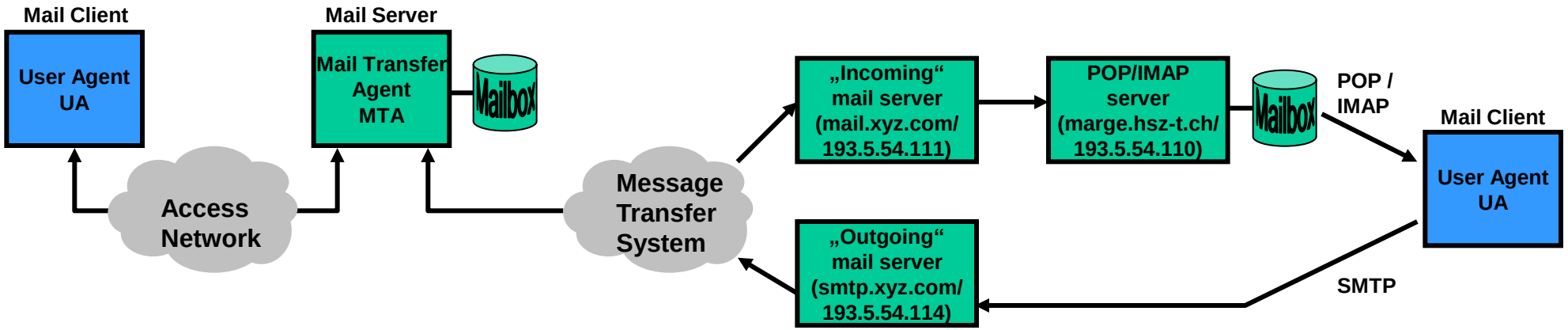
- SMTP is normally used for transferring emails between email servers.
- Receiver must be ‚online‘ to receive email message
- Store-and-forward
- SMTP is extremely simple, ➔ very stable/reliable ➔ very popular
- No security/authentication (spam!)

SMTP extended commands:

ESMTP RFC1869 SMTP Service Extensions

Typical mail server setup:

Organizations often run 2 distinct SMTP servers.
One server is for incoming mails (from outside to local email account), the other is for outgoing mails (from local account to outside).



5. Email Retrieval with POP3 Post Office Protocol RFC1939 (1/3)

Example POP session with TELNET (1/2):

```
cmd>telnet pop.indigoo.com 110
S: +OK POP3 bart v2001.78 server ready
C: USER pegli
S: +OK User name accepted, password please
C: PASS ********
S: +OK Mailbox open, 2 messages
C: LIST
S: +OK Mailbox scan listing follows
S: 1 591
S: 2 608
S: .
C: RETR 1
S: +OK 591 octets
S: Return-Path: <someone@somewhere.com>
S: Received: from dclient80-218-75-247.hispeed.ch (dclient80-218-75-247.hispeed.ch [80.218.75.247])
S: by pop.indigoo.com (8.12.9+Sun/8.12.9) with SMTP id hB2M1JXI027862
S: for pegli@fhzh.ch; Tue, 2 Dec 2003 23:01:41 +0100 (CET)
S: Date: Tue, 2 Dec 2003 23:01:20 +0100 (CET)
S: From: someone@somewhere.com
S: Message-Id: <200312022201.hB2M1JXI027862@indigoo.com>
S: X-Authentication-Warning: pop.indigoo.com: dclient80-218-75-247.hispeed.ch [80.218.75.247] …
S: Subject: test mail 1
S: Content-Length: 29
S: Status:
S:
S: das ist ein test mail.
S: ende
S: .
```

5. Email Retrieval with POP3 Post Office Protocol RFC1939 (2/3)

Example POP session with TELNET (2/2):

```
C: RETR 2
S: +OK 608 octets
S: Return-Path: <unknown@somewhere.com>
<2. mail here>
S: .
C: LIST
S: +OK Mailbox scan listing follows
S: 1 591
S: 2 608
S: .
C: DELE 1
S: +OK Message deleted
C: DELE 2
S: +OK Message deleted
C: QUIT
S: +OK Sayonara
```

N.B.: POP over SSL: Use port 995 instead 110

5. Email Retrieval with POP3 Post Office Protocol RFC1939 (3/3)

POP commands:

```
POP3 Commands
  Description

USER
  Specifies the username.

PASS
  Specifies the password.

STAT
  Requests the mailbox status (number of
messages,
  size of messages).

LIST
  Lists an index of all messages.

RETR
  Retrieves the specified message.

DELE
  Deletes the specified message.

NOOP
  Does nothing („no operation“).

RSET
  Undeletes messages (rollback).

QUIT
  Commits changes and disconnects.
```

POP Summary:

- Protocol for email retrieval from server.
- Max. one session for email retrieval.
- Supports only 1 mailbox.
- Emails are stored on client (UA client) and deleted on server.
- POP3 only allows offline mode (mails downloaded to UA).
- POP3 allows basic user authentication (username/password).

P.S.: some POP3 implementations allow to leave the messages on the server.

6. Email Retrieval with IMAP4 Internet Mail Access Protocol RFC2060 (1/2)

Example IMAP session with TELNET:

```
cmd>telnet pop.hsz-t.ch 143
S: * OK [CAPABILITY IMAP4REV1 LOGIN-REFERRALS STARTTLS AUTH=LOGIN] bart IMAP4rev1 2001.315 at Tue, ...
C: a001 login pegli ******
S: a001 OK [CAPABILITY IMAP4REV1 IDLE NAMESPACE MAILBOX-REFERRALS SCAN SORT THREAD=REFERENCES ...
C: a002 select inbox
S: * 0 EXISTS
S: * 2 EXISTS
S: * 2 RECENT
S: * OK [UIDVALIDITY 1070318215] UID validity status
S: * OK [UIDNEXT 262] Predicted next UID
S: * FLAGS (\Answered \Flagged \Deleted \Draft \Seen)
S: * OK [PERMANENTFLAGS (\* \Answered \Flagged \Deleted \Draft \Seen)] Permanent flags
S: * OK [UNSEEN 1] first unseen message in /afs/hsz-t.ch/usr/pegli/mbox
S: a002 OK [READ-WRITE] SELECT completed
C: a003 fetch 1 full
S: * 1 FETCH (FLAGS (\Recent) INTERNALDATE " 2-Dec-2003 23:55:07 +0100" RFC822.SIZE 437 ENVELOPE ...
S: "someone" "somewhere.com")) ((NIL NIL "someone" "somewhere.com")) NIL NIL NIL NIL ...
S: a003 OK FETCH completed
C: a005 store 1 +flags \deleted
S: * NO Unsupported system flag: \deleted
S: * 1 FETCH (FLAGS (\Recent))
S: a005 OK STORE completed
C: a006 logout
S: * BYE bart IMAP4rev1 server terminating connection
S: a006 OK LOGOUT completed
```

N.B.: IMAP over SSL: Use port 993 instead 143

6. Email Retrieval with IMAP4 Internet Mail Access Protocol RFC2060 (2/2)

IMAP commands:

IMAP4 Commands
  Description

USER CAPABILITY
  Requests a list of supported functionality

AUTHENTICATE
  Specifies an authentication mechanism

LOGIN
  Provides username and password

SELECT
  Specifies the mailbox

EXAMINE
  Specifies mailbox in read-only mode

CREATE
  Creates a mailbox

DELETE
  Deletes a mailbox

RENAME
  Renames a mailbox

SUBSCRIBE
  Adds mailbox to active list

LIST
  Lists mailboxes

LSUB
  Lists subscribed mailboxes

STATUS
  Requests mailbox status (number of messages, and so on)

APPEND
  Adds a message to the mailbox

CHECK
  Requests a mailbox checkpoint

CLOSE
  Commits deletions and closes mailbox

EXPUNGE
  Commits deletions

SEARCH
  Searches mailbox for messages meeting specified criteria

UNSUBSCRIBE
  Removes mailbox from active list

NOOP
  Does nothing

LOGOUT
  Closes the connection

FETCH
  Fetches parts of a specified message

STORE
  Changes data of specified messages

COPY
  Copies message to another mailbox

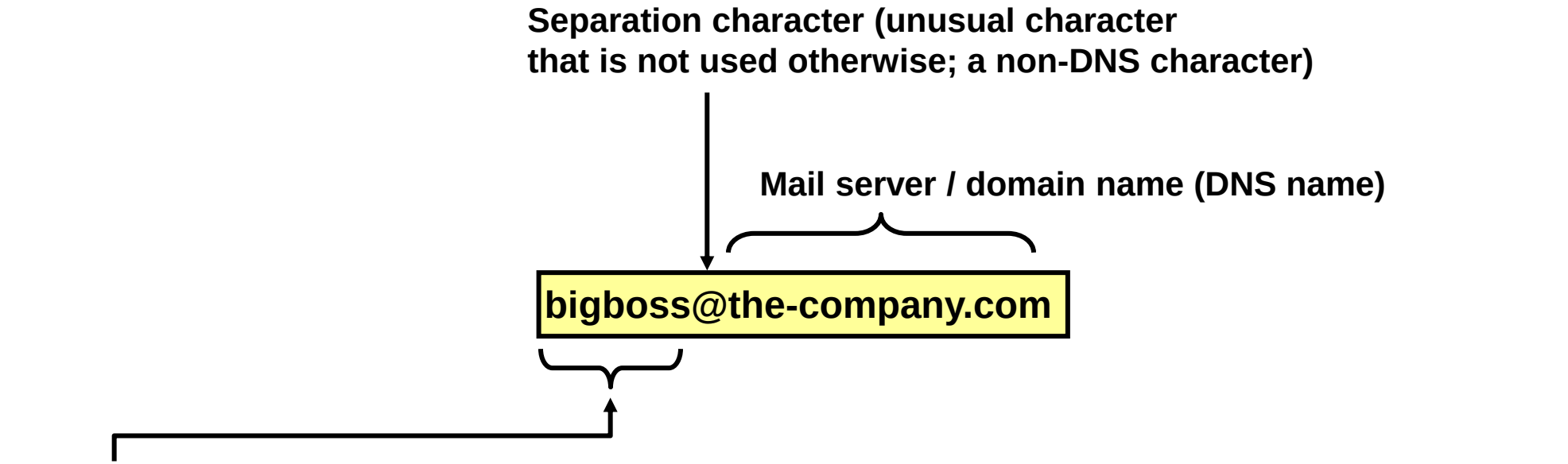
7. POP versus IMAP

POP and IMAP both serve the same purpose.
IMAP, however, has some notable differences as shown in the following table:

Feature	POP3	IMAP
Where is protocol defined?	RFC 1939	RFC 2060
Which TCP port is used?	110	143
! → Where is e-mail stored?	User's PC	Server
! → Where is e-mail read?	Off-line	On-line
Connect time required?	Little	Much
Use of server resources?	Minimal	Extensive
! → Multiple mailboxes?	No	Yes
Who backs up mailboxes?	User	ISP
Good for mobile users?	No	Yes
User control over downloading?	Little	Great
Partial message downloads?	No	Yes
Are disk quotas a problem?	No	Could be in time
Simple to implement?	Yes	No
Widespread support?	Yes	Growing

8. Email Address

Email addresses identify the sender and receiver of mail messages.



9. MIME Multipurpose Internet Mail Extensions (1/10)

SMTP RFC821 and mail format RFC822 define that envelope, header and body of message be in ASCII = 7Bit code (0-127).

Problem:

ASCII does not allow sending the following characters and alphabets:

a. **Transfer of binary data (where all 256 values of bytes used, e.g. 10110101)**
b. **Non-English text with accents ÅÇÊÑûĉ**
c. **Non-Latin alphabets (Chinese, Russian …)**

Mail infrastructure (mail proxies, servers, relays) often still supports only ASCII transfer.

Solution:

MIME allows sending non-ASCII data while preserving RFC822 message structure.
Thus MIME is downward compatible (ASCII-only mail infrastructure can be used).

9. MIME Multipurpose Internet Mail Extensions (2/10)

MIME extension fields:

MIME (RFC1521) defines new message header fields and adds a structure to the message body.

Email client (UA) must be enhanced to "understand" MIME format. If not the MIME message is displayed as text (as without MIME), but the actual message body is still readable.

Header fields for MIME

Header	Meaning
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Type and format of the content

The presence of the field `MIME-Version:` indicates that the mail message has MIME content.

9. MIME Multipurpose Internet Mail Extensions (3/10)

MIME extension fields:

MIME types (RFC2045)

Type	Subtype	Description
Text	Plain	Unformatted text
	Enriched	Text including simple formatting commands
Image	Gif	Still picture in GIF format
	Jpeg	Still picture in JPEG format
Audio	Basic	Audible sound
Video	Mpeg	Movie in MPEG format
Application	Octet-stream	An uninterpreted byte sequence
	Postscript	A printable document in PostScript
Message	Rfc822	A MIME RFC 822 message
	Partial	Message has been split for transmission
	External-body	Message itself must be fetched over the net
Multipart	Mixed	Independent parts in the specified order
	Alternative	Same message in different formats
	Parallel	Parts must be viewed simultaneously
	Digest	Each part is a complete RFC 822 message

9. MIME Multipurpose Internet Mail Extensions (4/10)

MIME message structure:

Simple MIME message

```
From: elinor@abcd.com
To: carolyn@xyz.com
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@abcd.com>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

This is the preamble. The user agent ignores it. Have a nice day.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/enriched

Happy birthday to you
Happy birthday to you
Happy birthday dear <bold> Carolyn </bold>
Happy birthday to you

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
        access-type="anon-ftp";
        site="bicycle.abcd.com";
        directory="pub";
        name="birthday.snd"

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

Boundary preceded by 2 dashes

MIME part

MIME part

Last boundary followed by 2 dashes

9. MIME Multipurpose Internet Mail Extensions (5/10)

Transfer Encoding (1/4):

Problem:
Not all mail transfer systems were capable of handling non-ASCII data (Unix systems). Some could only deal with ASCII (7-bit character) data.

Solution:
Convert non-ASCII data into ASCII strings for transmission, convert back into original data at receiver.

→ How to convert (encode) 8Bit data (text, binary) into 7-bit ASCII (where MSB =0)?

A. Base64 Encoding (="ASCII armor")
➔ Inflates data by 30%

B. Quoted Printable:
➔ More efficient for data that contains only few non-ASCII bytes, (adds only additional data for bytes that have MSB=1)

C. UUencode (UNIX)

D. No encoding at all, just send as binary data.

9. MIME Multipurpose Internet Mail Extensions (6/10)

Transfer Encoding (2/4):

Base64 Encoding:

Example: `10010101 11011100 00111011 01011000`

1. **Add 0x00 to get multiple of 3 bytes (padding):**
   `10010101 11011100 00111011 01011000 00000000 00000000`
2. **Take 24 Bit (3Bytes) and split into 4*6 Bit chunks:**
   `100101 011101 110000 111011 010110 000000 000000 000000`
3. **Convert 6 Bit values with Base64 (below):**
   Translate trailing artificial 0x00 bytes to "=" (only the ones that do not contain original data)
   Result: `ldw7 WA==`

Base64 translation table

Value (hex)	Char	Value (hex)	Char	Value (hex)	Char	Value (hex)	Char
0x00	A	0x10	Q	0x20	g	0x30	w
0x01	B	0x11	R	0x21	h	0x31	x
0x02	C	0x12	S	0x22	i	0x32	y
0x03	D	0x13	T	0x23	j	0x33	z
0x04	E	0x14	U	0x24	k	0x34	0
0x05	F	0x15	V	0x25	l	0x35	1
0x06	G	0x16	W	0x26	m	0x36	2
0x07	H	0x17	X	0x27	n	0x37	3
0x08	I	0x18	Y	0x28	o	0x38	4
0x09	J	0x19	Z	0x29	p	0x39	5
0x0A	K	0x1A	a	0x2A	q	0x3A	6
0x0B	L	0x1B	b	0x2B	r	0x3B	7
0x0C	M	0x1C	c	0x2C	s	0x3C	8
0x0D	N	0x1D	d	0x2D	t	0x3D	9
0x0E	O	0x1E	e	0x2E	u	0x3E	+
0x0F	P	0x1F	f	0x2F	v	0x3F	/

9. MIME Multipurpose Internet Mail Extensions (7/10)

Transfer Encoding (3/4):

Base64 Decoding:

Example: `ldw7 WA==`

1. Translate characters back to 6 Bit values with Base64 translation table (,=, becomes 0x00):
   `100101 011101 110000 111011 010110 000000 000000 000000`
2. Re-group to 8 Bit chunks:
   `10010101 11011100 00111011 01011000 00000000 00000000`
3. Strip trailing 0x00's:
   Result: `10010101 11011100 00111011 01011000`

Base64 translation table

Value (hex)	Char	Value (hex)	Char	Value (hex)	Char	Value (hex)	Char
0x00	A	0x10	Q	0x20	g	0x30	w
0x01	B	0x11	R	0x21	h	0x31	x
0x02	C	0x12	S	0x22	i	0x32	y
0x03	D	0x13	T	0x23	j	0x33	z
0x04	E	0x14	U	0x24	k	0x34	0
0x05	F	0x15	V	0x25	l	0x35	1
0x06	G	0x16	W	0x26	m	0x36	2
0x07	H	0x17	X	0x27	n	0x37	3
0x08	I	0x18	Y	0x28	o	0x38	4
0x09	J	0x19	Z	0x29	p	0x39	5
0x0A	K	0x1A	a	0x2A	q	0x3A	6
0x0B	L	0x1B	b	0x2B	r	0x3B	7
0x0C	M	0x1C	c	0x2C	s	0x3C	8
0x0D	N	0x1D	d	0x2D	t	0x3D	9
0x0E	O	0x1E	e	0x2E	u	0x3E	+
0x0F	P	0x1F	f	0x2F	v	0x3F	/

9. MIME Multipurpose Internet Mail Extensions (8/10)

Transfer Encoding (4/4):

Quoted Printable Encoding:

Rule #1:
Any byte can be represented by the string “=xx” where xx is the string representation of the 8Bit byte in hexadecimal.
Rule #2:
Bytes with MSB=1 (non-ASCII) must be encoded by string “=xx”.
Rule #3:
Equal sign is encoded as “=3D”.

E.g.: 10010101 becomes „=95“

Quoted Printable Decoding:

Translate each occurrence of “=xx” back into the original byte.

9. MIME Multipurpose Internet Mail Extensions (9/10)

Example Email/MIME session with TELNET (1/2):

```
cmd>telnet smtp.fhzh.ch 25
S: 220 smtp.indigoo.com ESMTP Sendmail 8.12.9+Sun/8.12.9; Tue, 2 Dec 2003 21:52:55 +0100 (CET)
C: HELO pegli
S: 250 smtp.indigoo.com Hello dclient80-218-75-247.hispeed.ch [80.218.75.247], pleased to meet you
C: MAIL FROM: someone@somewhere.com
S: 250 2.1.0 someone@somewhere.com... Sender ok
C: RCPT TO: pegli@fhzh.ch
S: 250 2.1.5 pegli@fhzh.ch... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: <enter mail header here>
C: <enter MIME content here>
C: .
S: 250 2.0.0 hB2KqtXI020827 Message accepted for delivery
```

9. MIME Multipurpose Internet Mail Extensions (10/10)

Example Email/MIME session with TELNET (2/2):

Sample MIME RFC822 mail with HTML and attachment:

```
From: Peter Egli <pegli@fhzh.ch>
Message-ID: <001d01c3ba87$c9dd4a20$4b2010ac@pacific>
To: <pegli@fhzh.ch>
Subject: MIME Demo Mail with HTML and Attachment
Date: Thu, 4 Dec 2003 17:58:00 +0100
MIME-Version: 1.0
Content-Type: multipart/mixed;
  boundary="----=_NextPart_000_0015_01C3BA90.283F4D40"
Content-Length: 1962
Status:

This is a multi-part message in MIME format.

------=_NextPart_000_0015_01C3BA90.283F4D40
Content-Type: multipart/alternative;
  boundary="----=_NextPart_001_0016_01C3BA90.283F4D40"


------=_NextPart_001_0016_01C3BA90.283F4D40
Content-Type: text/plain;
            charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

------=_NextPart_001_0016_01C3BA90.283F4D40
Content-Type: text/html;
            charset="iso-8859-1"
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <META http-equiv=3DContent-Type content=3D"text/html; =
     charset=3Diso-8859-1">
    <META content=3D"MSHTML 6.00.2716.2200" name=3DGENERATOR>
    <STYLE></STYLE>
  </HEAD>
  <BODY bgColor=3D#ffffff>
    <A href=3D"http://www.fhzh.ch">www.fhzh.ch</A>
    <DIV> </DIV>
  </BODY>
</HTML>

------=_NextPart_000_0015_01C3BA90.283F4D40
Content-Type: text/plain;
            name="Install.log"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: attachment;
            filename="Install.log"

This is the text in the attachment. Everything from
here up to the next boundary is attachment text.

------=_NextPart_000_0015_01C3BA90.283F4D40--
```