1 Message Handling

1.1 Message Handling Overview

1.1.1 Same time and different time communication

Message handling systems (MHS systems) are systems for people to send messages to each other. (Some messages are sometimes sent to or from applications without human intervention, but the main functionality is messaging between people.) The table below gives an overview of some major categories of MHS systems:

	All participants are active at the same time.	Different participants can read and write at different times.
Non-computer equivalent	*Face-to-face meeting, telephone call.*	*Letter, newspaper.*
Two or very few participants.	Simple chat systems, instant messaging systems.	Ordinary e-mail.
Communication within groups of people.	More advanced chat systems.	E-mail with mailing lists. Usenet News, forum systems.

There is no absolute sharp limit. Ordinary e-mail with multiple recipients is often used to communicate in small groups. And some chat systems contain archiving, so that those not present can read the discussion at a later time. There is, however, certain design features which are more important for the different categories.

For *same-time* messaging, it is important that everything said or written is immediately shown to the other participants. Some e-mail systems have an alert facility, where the recipient is reminded by a beep, whenever a new message arrives. Such e-mail systems can be used for almost same-time communication. But most textual same-time communication is done through specially designed chat or instant messaging systems. There are usually separate panes for reading and writing, sometimes the full discussion scrolls in a combined window for all participants, sometimes (only with very few participants) there is a separate window showing what each participant has written.

For *different-time* messaging, the set of messages which has been read and not read at a certain time will be different for each participant. People usually want to read only unseen messages, but they also want to be able to go back and reread old messages. Because of this, such systems have data bases, where messages are stored, and this data base knows what each user has seen and not seen and provides facilities for rapidly scanning unseen messages and searching for old messages. This can either be a personal data base with copies of all the messages for each recipient, or joint data bases, where the text of the messages need only be stored once, even if they have multiple recipients.

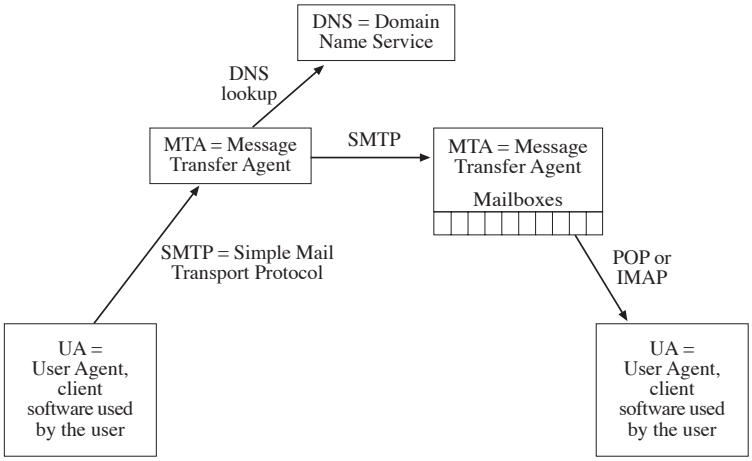
With *different-time* messaging, some people will get more messages than they can comfortably read. It is easier for them to handle the message load, if the software gives them facilities to organize the message data base. Typical such organization tools sort the messages from each discussion group into a separate folder, and sort messages in a thread (messages

which are direct or indirect replies to each other) together in some ways. The application layer standards have special facilities to support such functions.

With *same-time messaging*, it is important that messages are transmitted fast. The store-and-forward methods used by ordinary e-mail is too slow, but some systems for same-time messaging use their own kind of fast store-and-forward transmission. Particular for same-time messaging is that, since messages are written and arrive asynchronously, there can be dead times when the recipient receives nothing. It is, therefore, useful to have a reminder system, perhaps using a sound signal, when a new message arrives. There is also a tendency, in same-time messaging, that the on-going conversation handles more than one topic at the same time. People unaccustomed to chat systems react against this, and sometimes requre “one subject at a time”, just as at face-to-face meetings following an agenda. However, one subject at a time is not the most efficient use of the time. Because people write slower than they read, and because dead times occurs when you are waiting for a reply to a message, the time is used more efficiently if you allow multiple topics in parallel.

Two main variants of the user interface are common in chat systems. One variant has a separate window for each participant, showing what that person has written. The other has a single sequential window all participants of the same chat channel, listing all messages in cronological order. A *channel* is a group of people who are discussing with each other in a chat system.

1.2 Protocols Used for Message Transport



Many application layer protocols are commonly used for message transport. Here are some of the most commonly used:

- ♣ The DNS is used to look up the mail host for the recipient, and then in a second step to look up the IP address of the MTAs of the receiving MTA.
- ♣ SMTP is used to transport the message to each handling MTA.
- ♣ POP or IMAP is used to download the message from the last MTA to the recipient UA.
- ♣ MSGFMT (previously known as RFC822) and MIME are used to format the content of a message.
- ♣ NNTP for distribution of Usenet News Articles.

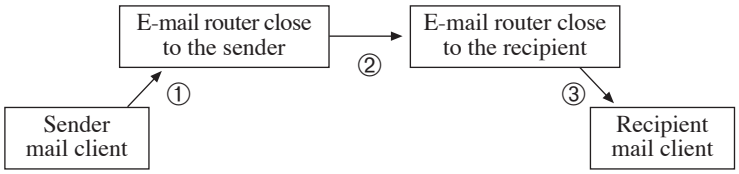
In addition to this, messages may also be conveyed through *usenet news*, which has its own

set of protocols, partially based on the email protocols, but often with their own variations, and messages may be conveyed through *instant messaging* protocols.

Note that the receiving user's mailbox is usually on the last MTA, and not in the personal computer of the recipient. This is important, because personal computers are not always online. Messages can thus be delivered to a mailboxes, even if the actual recipients are travelling, ill, playing games or otherwise not connected to the Internet.

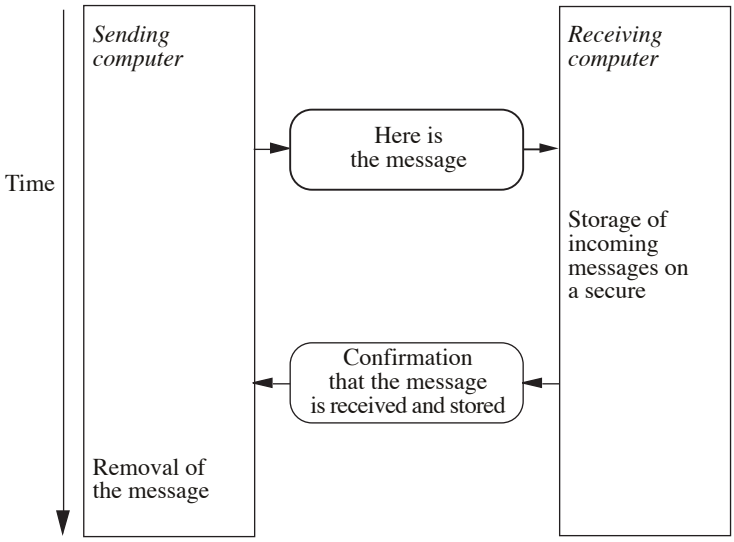
1.3 E-mail: Message Transport

Typical for e-mail and other non-simultaneous message systems is that the messages are copied, often several times, between different mail servers on their way from sender to recipient. The picture below shows a simple and common route which a message may take from sender to recipient:



When senders submit messages, their mail clients connect to nearby e-mail routers. The whole message is then copied ① from sender to mail router. When this copying is ready, this mail router takes over responsibility for delivering the message to its recipients:

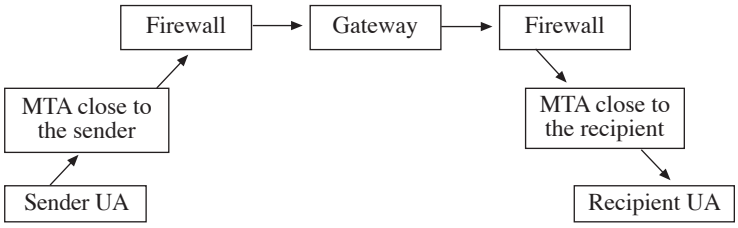
When the receiving computer confirms reciept of the message, the sending computer is no longer involved. In the second step, the mail router close to the sender connects to a mail router close to the recipient. Again, the whole message is copied ②, and responsibility is then transferred to the e-mail router close to the recipient. In the final step, the message is copied ③ to the recipient mail client.



In e-mail terminology, the client of the sender and the recipient are often called *User Agents* (UA), and the e-mail routers are often called *Message Transfer Agents* (MTA).

The transport of a message does not always pass exactly two MTAs as in the figure above.

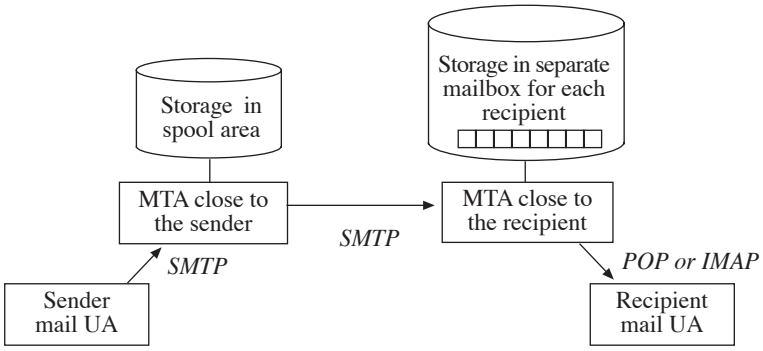
If both sender and recipient work at the same place, only one MTA may be needed. And in a complex networking situation, more than two MTAs can be used, like in the figure below:



A firewall is a security device, and a typical function in it may be to scan for viruses in the message. A gateway is a device which moves messages between two networks using different e-mail protocols, for example from an Internet mail network to an X.400 mail network. (X.400 is a mail protocol developed by the Telecom companies, it is today only used in certain local areas.)

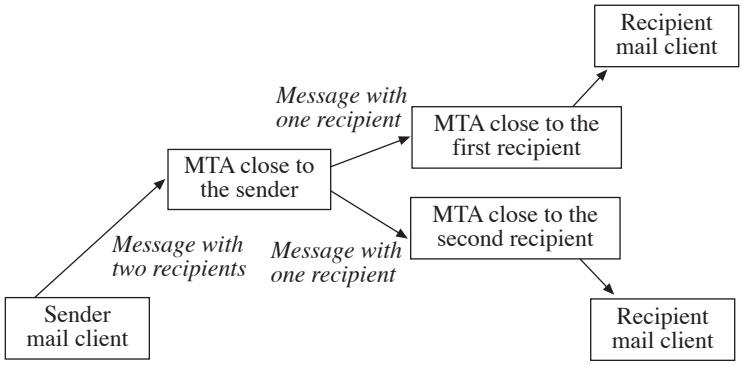
When a message is passed between two agents, a protocol is needed. The most common protocol for passing messages between agents i SMTP. SMTP is almost universally used for all mail transport except the transport from the last MTA to the recipient UA. The reason for this is that SMTP is mainly designed for sender-controlled transmission. In every step between two agents, it is the sending agent which controls the transmission and decides when and where to send a message. This is not suitable for recipient UAs, since they often reside on personal computers, which are not always connected to the network and willing to receive messages. Thus, for the final step from the last MTA to the recipient UA, a protocol is needed where the recipient has more control of when to receive a message. The two most common protocols for this are POP and IMAP. In some cases, the last MTA and the recipient UA run on the same computer or on two computers sharing a common file storage. In that case, the protocol for the last step can just be simply file retrieval. Another important difference between SMTP and the last-step delivery protocols is that identification (usually with a password) is needed to stop messages being retrieved by other than its intended recipients.

An MTA which receives a message will usually store it in a so-called spool area. This is a file storage area for temporarily storing files, and where there is some process, the spool process, which goes through the spool areas and forwards messages from them. An exception to this is the last MTA before delivery to the final recipient UA. That MTA usually stores messages in mailboxes, separate areas for each recipient. This means that when a UA connects to an MTA to download mail, all mail for that particular UA is stored in a special place and can easily be located. See the figure below:

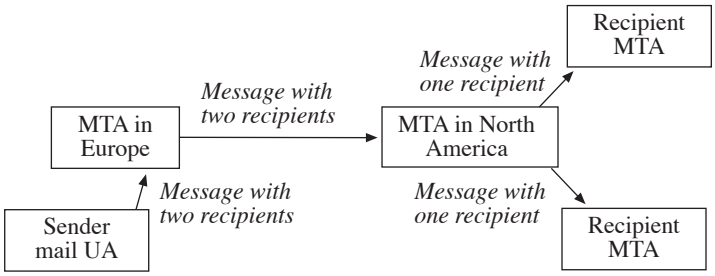


If a message is sent with more than one recipient, the message may be split into separate

copies by one or more MTAs, as shown in the figure below:



The advantage with splitting is that the same message need not be transported more than once before the splitting. This can also be used to reduce transport costs across large distances. If a sender in Europe sends a message to two or more recipients in North America, only one copy might be copied across the expensive Atlantic cables as shown in the figure below:



A problem with this, however, is that most MTAs are not willing to handle mail, unless either the recipient or the sender is local to the MTA. Thus, the saving shown above requires an agreement with the MTA which splits the message after transport across the Atlantic. This was not always so. In the beginning of the 1990-s, most MTAs were willing to forward mail for any recipient. The reason why this was abolished in the middle of the 1990-s was that spammers used this feature to get foreign MTAs to help them split mail to millions of recipients. Some so-called experts claimed that spamming could be stopped by forbidding splitting of mail by other than the MTA of the sender or the recipient. They enforced their view by implementing a program which scanned all MTAs everywhere, checking that they did not allow foreign splitting, and sending angry letters to non-conforming MTA administrators (postmasters) threatening to stop receiving mail from them unless they stopped splitting. This is an interesting example of how the Internet is regulated in dubious ways by pseudo-police-authorities. Spamming could be counteracted more efficiently using other methods than this. In fact, there is no proof that spamming has in any way been reduced by this method.

flame

Here I am flaming: giving my personal opinion on a controversial issue. Other experts have different opinion on this.

1.4 Use of the DNS for mail transport

E-mail has its own special way of using the DNS. The DNS can actually be seen as two different data bases, one for e-mail and one for all other usage. The special records for e-mail are called MX (mail exchange) records.

If an MTA receives a message for a recipient with the e-mail address mary@bar.net, it will connect to the DNS and ask for the MX record for the domain “bar.net”. It may then be told that the MX record for “bar.net” refers to the domain name “mail.bar.net”. It will then make a second ordinary (not MX) DNS lookup to find the IP number for “mail.bar.net,” and it will then forward the message to the SMTP port at this IP number.

A special facility of the DNS, when used for e-mail, is that a lookup for the MX record for a domain (like “bar.net” in the example) may return more than one result. It may return a list of domains. The reason for this is to increase reliability, by having more than one MTA which is willing to handle mail for a particular recipient. An organisation may have one primary mail MTA and another secondary mail MTA to use if the primary mail MTA is not in operation. A common usage of this is that if each department in an organisation has different MTAs, the organisation may have a common master MTA, which is used if one of the department MTAs is out of operation.

There is, however, a risk with this technique. Suppose that an organisation **foo.bar** has three different MTAs, **mail1.foo.bar**, **mail2.foo.bar** and **mail3.foo.bar**. Suppose that a message arrives at **mail3.foo.bar**, but that the malbox of the recipient actually resides in **mail1.foo.bar**. The MTA at **mail3.foo.bar** will then use the DNS to look up the MX record for foo.bar, and may get a list of **mail1.foo.bar**, **mail2.foo.bar** and **mail3.foo.bar**. It might then transfer the message to **mail2.foo.bar**. The MTA at **mail2.foo.bar** will make a new MX record look up, get the same result, and transfer the message back to **mail3.foo.bar**. In this way, an infinite loop may occur with the message being sent back and forward between **mail2.foo.bar** and **mail3.foo.bar**.

To avoid such loops, the DNS has a special facility. It stores, with every MX record, a priority. In the example above, a DNS lookup for **foo.bar** may return the three MTA names **mail1.foo.bar**, **mail2.foo.bar** and **mail3.foo.bar**. But they might each have a priority, for example:

mail1.foo.bar	0
mail2.foo.bar	10
mail3.foo.bar	20

There is then a rule, that an MTA should never transport a message from an MTA with a lower MX record priority value to an MTA with a higher MX record priority value. Thus, **mail2.foo.bar** can never transport the message to **mail3.foo.bar**, and no loop will occur.

There is then a rule, that an MTA should never transport a message from an MTA with a lower MX record priority value to an MTA with a higher MX record priority value. Thus, **mail2.foo.bar** can never transport the message to **mail3.foo.bar**, and no loop will occur.

1.5 Mailbox names

Senders and recipients of e-mail are called *mailboxes*. Note that the recipient mailbox is a box on the last MTA, from which the recipient can download messages. The mailbox is not a storage area on the recipient computer, unless the recipient computer and the last MTA is the same computer (See the figure on page 3).

A mailbox in SMTP usually has two names, a user-friendly name and an e-mail address. For example, my user-friendly name is “Jacob Palme” and my e-mail address is “jpalme@dsv.su.se”. The two names are often written in sequence in the format shown by these examples:

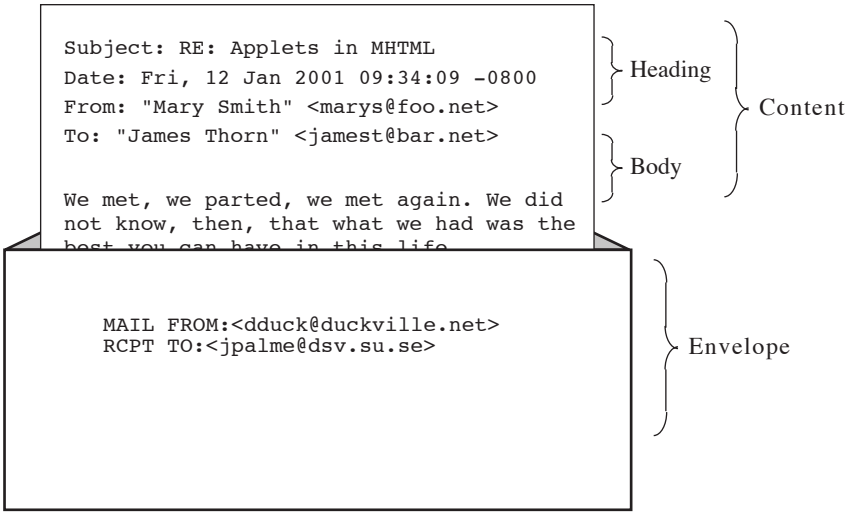
Jacob Palme <jpalme@dsv.su.se>

"Nils B. Frändén" <nfranden@foo.bar>

If the user-friendly name contains certain special characters, including periods followed by spaces and non-latin letters, it must be enclosed in double quotes as shown above. However, this rule is often broken, so it is advisable to write a mail program so that it accepts user-friendly names without double-quotes.

As is shown by the examples above, the user-friendly name is not globally unique – two different people may have the same user-friendly name. The e-mail address must be globally unique (otherwise it would not be possible to know to which mailbox to deliver the mail), it consists of a local mailbox name, the “@” character and a globally unique domain name. (Some servers accept mail with an incomplete domain name, if the address is a local user to the domain served by the server, but this practice is not recommended. It is much better to always use globally unique e-mail addresses, because then the address of a user will be the same independently of where it is sent from.)

1.6 Message format



A message being sent from sender to recipient consists of an *envelope* and a *content*. The content consists of a *message heading* and a *body*.

The *envelope* contains information used or generated during the transport of the message, such as the e-mail address of the recipient and the e-mail address of the sender. The sender's address is needed to be able to send a non-delivery notification if the message cannot be delivered to the recipient.

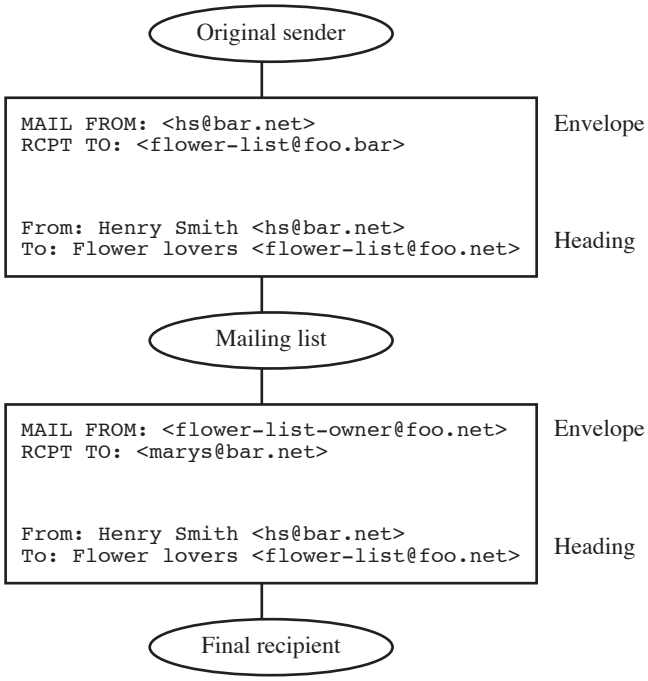
The *content* contains the information which is sent from the original author to the final recipient. It is, in theory, not meant to be changed during transmission. In reality, it is sometimes changed. Certain information which logically belongs to the envelope is in reality

put in the heading. And the content is sometimes transformed, for example between two different encoding formats, such as 8BIT and Quoted-Printable.

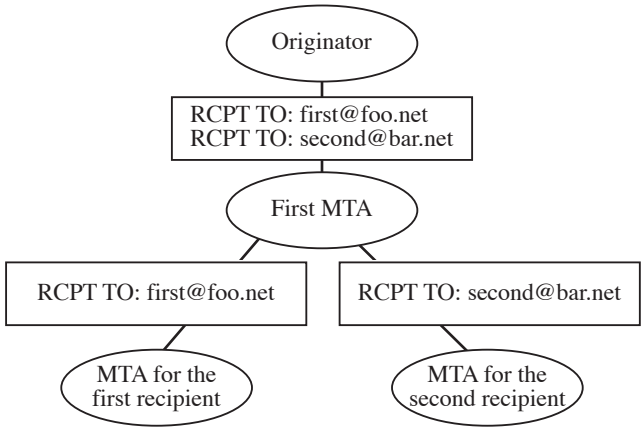
The *body* contains any kind of information, such a text, pictures, sound, video and/or file attachments.

Some information is specified both on the envelope and in the heading, such as the sender and the recipient. This information is, however, not always the same. The envelope shows information pertaining to a particular transmission or step in transmission. The heading shows information created by the sender and intended for the final recipient.

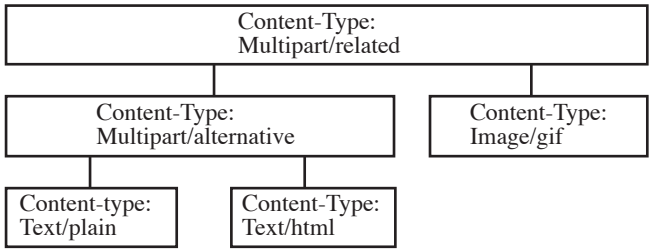
Example 1: If a person sends a message to a mailing list, and the mailing list forwards the message to the members of the list, then the heading contains the original sender in the "From:" field and the name of the mailing list in the "To:" field. The envelope contains different information when transporting the message from the original sender to the mailing list, and when transporting the message from the mailing list to the final recipients, as is shown by the figure below:



Example 2: If a message is split into copies to be delivered to different recipients, then each copy will on the envelope contain the address of the recipient for this copy. A message may, for example, first be sent with two recipient from the sender UA to the first MTA. The first MTA may then split the message into two copies for the two recipients. The envelope will then in the second step of transmission contain only the recipient for this transmission, see the picture below:



The body can consist of several body parts, for example a text and an attachment or a HTML-formatted text and embedded pictures. Each body part can also consist of several body parts. The figure below shows a complex message with body parts inside body parts; this particular structure is very common:



When the body consists of several parts, each part has its own *content-heading*. A content-heading contains information about the content part, such as *content-type* and *encoding method*. The set of header fields allowed in a content-heading is a subset of the set of headers allowed in a message-heading. All the header fields allowed in a content-heading have names beginning with the string Content- such as “Content-Type”, “Content-Language”, etc.

1.7 E-mail: Message Heading

1.7.1 The Internet Message Format

The format of the messages themselves (corresponding to the P2/P22 protocols of X.400) is specified in RFC 822 [17], RFC 1036 [19], RFC 1123 [20], and the multi-purpose Internet mail extension (MIME) standards [21, 22].

The figure below shows an example of a message heading according to the RFC 822 standard. Note that what is shown is not some legible print-out of the heading, but the actual contents, since RFC 822 uses readable IA5 characters in the heading.

```
From comp.protocols.iso.x400-outbound-request@ics.uci.edu Wed
      Nov  4 04:16 MET  1992
Received:   from sunic.sunet.se by heron.dafa.se (16.6/SiteCap-3.0)
      id AA09605; Wed, 4 Nov 92 04:16:24 +0100
Received:   from ics.uci.edu by sunic.sunet.se (5.65c8/1.28)
      id AA25221; Wed, 4 Nov 1992 04:15:03 +0100
```

```
Received:   from ics.uci.edu by q2.ics.uci.edu id aa14794; 3 Nov 92
      13:47 PST
Received:   from USENET by q2.ics.uci.edu id aa14789; 3 Nov 92 13:46
      PST
From: "Paul.Rarey" <Paul.Rarey@ssf-sys.dhl.com>
Subject:    Re: X400 address
Message-ID: <921103134349.16483@maverick.ssf-sys.DHL.COM>
Encoding:   35 TEXT, 12 TEXT SIGNATURE
X-Mailer:   Poste 2.0
Date: 3 Nov 92 21:46:13 GMT
To:   mhsnews@ics.uci.edu,
      Piet Beertema <mcvax!cwi.nl!piet@uunet.uu.net>
cc:   ifip65@ics.uci.edu

... Text of the message ...
```

Example of an RFC 822 message heading.

User names in RFC822 can have both a user-friendly name and an e-mail address (see page 8).

RFC 822 specifies a heading format, which is intended to be readable for both humans and computers. It is used for communication between the sender and the recipient and is not used during transmission. The most important fields in Internet mail headings are:

Received: Contains a trace of the hosts through which a message has passed. It is mainly used when investigating problems with the mail transfer. Such a trace could be used for loop control (as in X.400) but in practice usually is not. Many UA softwares suppress these lines when showing the header to their users.

From: The e-mail address of the author of the message.

To:, Cc: and Bcc: The recipient(s). Note that this is not the recipient(s) to be used when delivering the mail (those are given in the SMTP protocol). This is information to be read by the recipient(s) or their mail program. It can even contain names which are not proper e-mail addresses.

Message-ID: A globally unique identity code for the current message, which can be used in the “Reply-To” field to eliminate duplicates of the same message arriving via different routes.

Reply-To: Used to indicate that personal replies to a message are to be sent to someone else than the name in the “From” field. Note that the name in this field should not be used for “group replies,” that is, replies intended for the group of recipients or the mailing list that received the In-Reply-To message.

Followup-To: Similar to “Reply-To” but used for group replies. The value must be a usenet newsgroup name, not an e-mail address. (This is not part of RFC 822 it is defined in the Usenet News standard, RFC 1036 [19].)

In-Reply-To: Used to coordinate replies with the original message. An intelligent mail software can use this field to help the user find the original message of a reply.

References: Similar to “In-Reply-To.” It is used mainly in Usenet News to indicate references between postings to newsgroups.

Date: The date, time, and time zone when the message was sent.

Subject:: A title line describing the topic of the message.

The RFC 822 heading often contains additional fields not defined in the RFC 822 standard. Some of them may be defined in other standards, while others are not standardized at all. In the heading abpve, the fields “Encoding” and “X-Mailer” are not part of RFC 822.

The RFC 822 heading ends with a blank line, which marks the beginning of the text of the message.

1.7.2 References between Messages and Global Message IDs

There is often a need to let a message refer to previously sent messages: for example, in replies. It is an important facility for a recipient of a reply to be able to easily ask his electronic mail system for the message to which the reply replies. It is also useful for users of electronic mail systems to be able to browse through whole conversations. In order to be able to transfer a reply link between two messages which are sent at different times, it is useful if every message has a globally unique identifying code. This is a code which no other message has or is going to get.

E-mail has some kinds of identifying codes in the heading and sometimes also on the envelope. The code on the envelope, however, is shortlived, and long-lived references between messages are handled using the identifying code in the heading. A globally unique identifying code in the heading is usually produced by combining two subfields. One subfield is a valid electronic mail name (usually of the originator, but some other valid name can also be used) plus an identifying code which is unique relative to this electronic mail name. Since electronic mail names and addresses must be unique (otherwise it would not be possible to deliver a message to its recipient), this will then produce a globally unique identifying code for the message itself. Often, these parts are separated with an asterisk. Example: **200107081059*john@foo.bar**. The part before the asterisk is usually an encoding of the date and time when the message was sent in some format. Another variant is to use a sequential numbering of all messages sent by john@foo.bar. A disadvantage with a sequential number is that the last used number need not be stored, and if this stored number is lost, duplicate Message-IDs will not be created. Therefore, the date/time method is usually better.

E-mail has three different kinds of references between messages, all indicated by use of the Message-ID:

In-Reply-To: for ordinary replies.

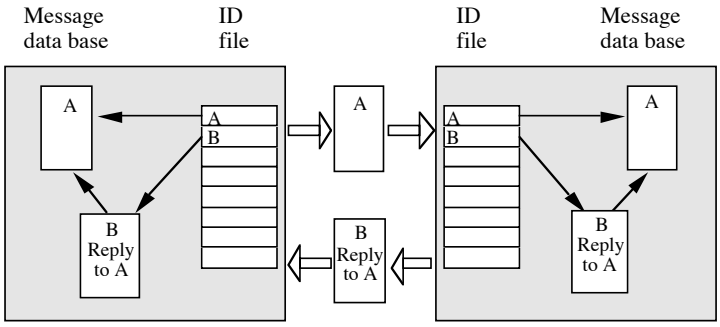
References: for other kind of references.

Supersedes: when the new message is a replacement of an old message, for example, a new version of a text under development or containing a copy of the original message with an error corrected. Supersedes is not yet standardized, but is commonly used in Usenet News, but not so often in e-mail.

A message cannot have an *In-Reply-To* reference to more than one previous message, while

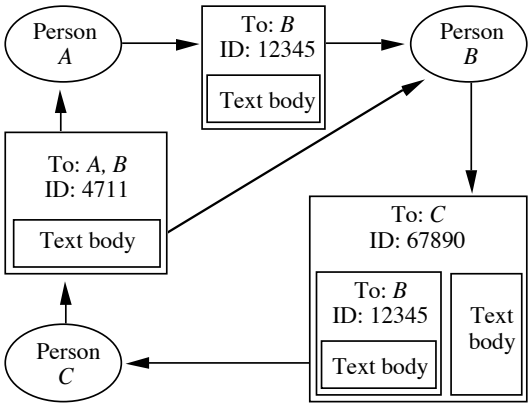
the number of messages it is *References:* or *Supersedes:* can be more than one. A consequence of this is that a conversation which is created using the *In-Reply-To* field will always have a tree structure, something which is not true for conversations created by *References:* and *Supersedes:*. Another consequence is that two different conversations can be merged into one if the *References:* and *Supersedes:* fields are used to create conversations. Such merging is not possible if only the *In-Reply-To* field is used.

The figure below shows how the Message-ID can be used to connect a reply to the original message, using a data base of Message-ID-s with pointers to the messages they refer to in the mailbox data base of both the sender and the recipient.



Use of Message-ID to correlate replies with the original message.

When you create a reply link on a message which contains forwarded body parts, you should carefully consider to which heading the reply link should refer. This is illustrated in the figure below.



Problem with references to the wrong Message-ID.

Person *A* writes a message to person *B*. *B* forwards the message to *C*, using the forwarding mechanism of including the whole text of the original message as a body part of the forwarded message. *C* then writes a reply to the original message and sends the reply to both *A* and *B*. It is very important for *C* to think carefully of whether to use the ID of the original message (12345, in the example) or the ID of the forwarding message (67890, in the example). This is important because *A* has never seen the forwarding message. This means that a reference in the reply, saying that it is a reply to 67890, will not be understood by *A*. Good message systems should be designed to protect users from such mistakes.

Of course, the intention of *C* may be to reply also or mainly to the text body which *B*

added when the message was forwarded to *C*. But if that is the case, *C* should either send the reply only to *B* or forward the whole message on to *A*, since a reply sent to *A* on the forwarded body part in 67890 will otherwise not be understood by *A*.

A person will often get several copies of the same message, forwarded by different users and distribution lists. It is an important service to the user that his electronic mail system be able to correlate these copies, so that the person will not have to read the texts more than once and so that the person can find all comments, appendices, and recipients of the message. Such correlation should also be done when the recipient receives the same message both directly and included as forwarded body parts in other messages. The globally unique Message-ID is the code which is used to provide such correlations. Note the use of the word "correlate," not the word "merge." Since all copies of the same message are not always identical, information can be destroyed if they are merged carelessly.

1.8 E-mail: Message Body

1.8.1 Types in Internet Mail

1.8.1.1 MIME Introduction

The Internet mail standards used before 1992 could handle only text in the 7-bit ASCII alphabet and did not allow the additional body-part type facilities available in X.400. However, in 1992, the Internet mail standards added facilities in an extension called MIME. MIME is defined in RFC 2045 [21] to RFC 2049 [22]. A good tutorial on MIME is given in [25]. There is an FAQ[2] on MIME [44] which includes a list of known MIME implementations.

MIME allows Internet mail to contain the following

- Multiple objects in one message.
- Unlimited line length and message length.
- Character sets other than IA5 (7-bit ASCII). In particular, the ISO 8859-1 and the ISO 10646 (Unicode) character set is important.
- Binary and application-specific files.
- Diagrams, pictures, voice, video, and multimedia in messages.
- References to files, which can be retrieved automatically through the net when the recipient wants to read the message. The contents of the file is thus not transported with the message.

---

2 FAQ (frequently asked question) is a document containing answers to often asked question in Internet discussion groups. There are many FAQs on various topics, and they often contain a lot of useful information.

With MIME, it is possible to have several body parts representing the same information in different formats. The recipient or his client software can then choose the version which it is best capable of displaying. For example, a recipient with an IA5-terminal can see a message in this format, while a recipient with an ISO 8859-1 terminal can see a better version of the message.

1.8.1.2 MIME Encoding of Data

There was a problem when MIME was defined: it was essential that existing e-mail systems be able to handle MIME messages in a reasonable way without modification. However, some previous Internet e-mail systems could not handle messages with 8-bit characters in the text or with unlimited line length, binary information, etc. in the body.

Because of this, MIME encodes the new body parts in a format which looks like 7-bit ASCII to old mail software. This is not a very neat solution, but it was necessary since no extension facility for body parts was included in the old Internet mail standards. MIME encodes additional information in either of two ways so that it looks like 7-bit ASCII. These two methods are called *base64* and *quoted-printable*.

Base64 uses four ASCII characters to represent three 8-bit bytes. The 24 bits of the three 8-bit bytes are split into four groups of 6 bits, and each such bit is encoded in a character set which has 64 different values. These 64 values are those characters in 7-bit ASCII which are least likely to be corrupted. (This is more fully described in the chapter about coding.)

Quoted-Printable represents those bytes, which have an exact correspondences in 7-bit ASCII, with their 7-bit ASCII values. Other characters are encoded as an "=" character, followed by two digits with the hexadecimal value of the byte. An exception is the "=" character itself, which is coded as "=3D."

1.8.1.3 Example of a Complete MIME-Encoded Message

The figure below shows a complete example of a MIME message as it is transmitted (not as it is shown to the user). This message contains two body parts. Each of the body parts contains the same text in ISO 8859-1.

```
Test message containing 8-bit characters.
AE=Ä, OE=Ö.
```

The two lines above are transmitted in both body parts in the MIME message below, but the text is encoded as quoted-printable in the first body part and as base64 in the second body part.

```
Return-Path: <jpalme@ester.dsv.su.se>
Date: Sun, 26 Sep 1993 18:49:01 +0100 (MET)
From: Jacob Palme DSV <jpalme@dsv.su.se>
Subject: A MIME message
To: Lars Enderin <larse@dialog.se>
Message-Id: <3.85.9309261822.A27024-0200000@ester>
Mime-Version: 1.0
Content-Type: MULTIPART/ALTERNATIVE; BOUNDARY="1430317162"

--1430317162
Content-Type: TEXT/PLAIN; CHARSET=ISO 8859-1
Content-Transfer-Encoding: QUOTED-PRINTABLE

Test message containing 8-bit characters.
AE=3D=80, OE=3D=85.
```

```
--1430317162
Content-Type: TEXT/PLAIN; CHARSET=ISO 8859-1
Content-Transfer-Encoding: BASE64

VGVzdCBtZXNzYWdlIGNvbnRhaW5pbmcgOC1iaXQgY2hhcmFjdGVycy4KQUU9
gCwgT0U9hS4K
--1430317162--
```

A complete MIME message.

As can be seen from the example in the figure above, quoted-printable has an advantage over base64 because a recipient whose mail system is not MIME compatible will still be able to understand much of the content, especially the ordinary English text. In the example, quoted-printable is also shorter: only 61 characters are required compared to 73 characters for base64. Base64 is, however, more suitable for pure binary text, such as a graphic bit image or an encoded sound.

In addition to base64 and quoted-printable, MIME also allows the encodings *7bit*, *8bit* and *binary*. These are not really encodings, since they represent uncoded data. Because of this, 8bit and binary should not be transmitted via mailers which are not MIME-compatible. To transport such data uncoded, extensions to the SMTP standard for message transport are also needed.

1.8.1.4 MIME Heading Extensions

MIME defines the following extended fields to the RFC 822 heading:

`Content-Type:`	Specifies the type and subtype of the data in the body.
`Content-Type: TEXT`	Specifies textual information in several different character sets.
`Content-Type:`	Specifies that the body contains more than one body part. Each body part can have additional heading fields at the beginning of the body part.
`Content-Type:APPLICATION`	For application-specific or binary data. Of special importance is `APPLICATION/POSTSCRIPT`.
`Content-Type: MESSAGE`	For an encapsulated mail message.
`Content-Type: IMAGE`	For a bitmapped picture using, for example, the graphics interchange format (GIF) or joint photographic experts group (JPEG) formats.
`Content-Type: AUDIO`	For sound. (Note that the word “voice” is used in X.400. However, X.400 voice is probably not intended to be restricted only to spoken sounds.)
`Content-Type: VIDEO`	For video using, for example, the motion picture experts group (MPEG) format. The video may, but need not, contain a sound track.
`Content-Transfer-Encoding:`	Specifies how the data is encoded. Encoding type may also be given in the content-type heading field.
`MIME-version:`	To indicate that MIME is used and which MIME version is used.
`Content-ID:`	ID code for the content. Can be used to allow one body part to refer to another body part in another message or for references between the body parts in a message.
`Content-Description:`	Contains a textual description of the content.

IANA (Internet Assigned Numbers Authority), the internet global registration authority, maintains a register of MIME content types. IANA requires a publicly available description of

the new format, or, alternatively, that public domain viewers for the new format are available. A consequence of this is that some commercial companies have been unwilling to to register the formats of their products. In such cases, a Content-Type EIGHTBIT has to be used instead. The disadvantage with this is that the receiving client software cannot automatically start a viewer for the new format when messages in that format arrives.

1.8.1.5 Multipart messages in MIME

The MULTIPART attribute in a MIME heading specifies that the body contains more than one body part. Each body part can have additional heading fields at the beginning of the body part, including recursive use of the multipart attribute to include MULTIPARTs as parts in higher level MULTIPARTs.

Of special interest are

`MULTIPART/MIXED`	for several parts of different types;
`MULTIPART/ALTERNATIVE`	for the same information in more than one encoding;
`MULTIPART/PARALLEL`	as mixed, but to be viewed at the same time (for example, voice in parallel with a drawing); and
`MULTIPART/DIGEST`	to indicate that this is a collection of several messages written by different authors.
`MULTIPART/RELATED`	several files, which are to be combined into one document or set of related documents. Used, for example, to send HTML, SGML or XML with pictures, applets, etc., as separate files.

1.8.1.6 The Multipart/Related Content Type

The Multipart/related content type is designed when you are sending several files, which are related by URL-links. It is used, for example, to send HTML, SGML and XML with embedded pictures or applets as separate files.

Each file is a separate body parts. Each body part is labelled by either Content-ID or Content-Location. The URL referring to the body part from another body part, is of the URL type "cid:" to refer to a Content-ID, or can be any kind of URL (absolute or relative) to refer to a Content-Location with the same content.

Example (abbreviated):

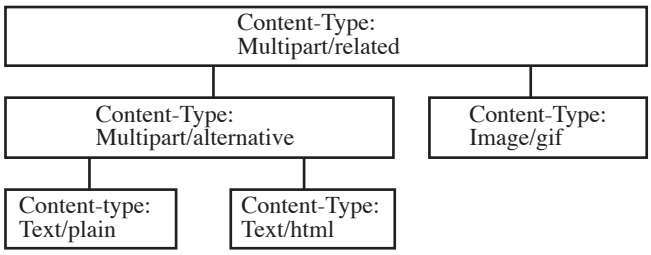
`Content-type: Multipart/related`	The compound object of the HTML text and the embedded message.

`Content-Type: Text/html`
 	
`<IMG SRC="cid:1*foo@bar.net>`
 	
`<IMG SRC="picture.gif">`
	The main text in HTL format.
	
	Link to an embedded image using a "cid:" type URL.
	Link to an embedded image using a relative URL.

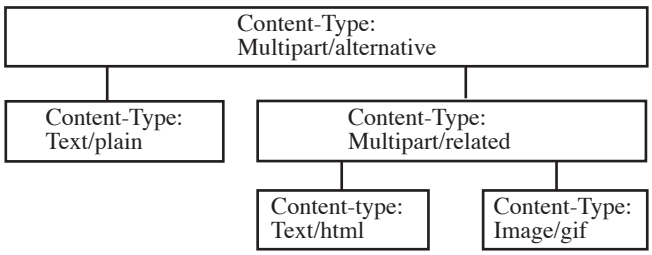
`Content-Type: Image/gif` `Content-ID: 1*foo@bar.net`	The first embedded image, identified by a Content-ID.
`Content-Type: Image/gif` `Content-Location: picture.gif`	The second embedded image, identified by a Content-Locdation URL.

Since some mailers do not support this, messages are usually sent using multipart/alternative, with plain text in the first branch and HTML in the second branch. This can be done in two ways:

1.8.1.6.1.1 With the multipart/alternative inside the multipart/related:



1.8.1.6.1.2 With the multipart/alternative outside the multipart/related:



Some mailers send messages using each of these methods, so a good mailer will have to be able to receive messages in both formats.

1.8.1.7 The Message Content Type

The MESSAGE content type is used to forward a message within another message. For example, you can forward a message you have received, as one body part, with your own comment as another body part. MESSAGE has the following subtypes:

- MESSAGE/RFC822 when you encapsulate a normal Internet mail message, formatted according to the RFC822 or MIME formats.
- MESSAGE/PARTIAL if this is one part of a large message, which you have split into several smaller messages during transport. Information is in this format included to help the receiving mail client merge the parts together into a complete message again.
- MESSAGE/EXTERNAL-BODY; ACCESS-TYPE= to send, instead of a message, just a reference to where the message can be retrieved. ACCESS-TYPE can indicate various ways of retrieving the actual content, like various variants of FTP etc.

1.8.1.8 Mime Heading Character Sets

MIME allows extended character sets also in message headings. The encoding is rather ugly and some implementations do not support it.

1.8.2 In-line or attachments

In the user interface, a MIME message can either be displayed as a sequence of text passages and images, or MIME can be used to send attachments to a primary textual message. The attachments are then not viewed unless the user explicitly asks for them. In practice, the latter

usage has been most common. RFC 1806 specifies an e-mail header which can be used to indicate, separately for each body parts, whether this body part is to be shown inline or as an attachment. The format is `Content-Disposition:` followed by either the word `inline` or the word `attachment`. A `file name` parameter can also be included, indicating a recommended file name if the attachment is stored in a file.

Note that if the `Content-Type: Multipart/related` is used to send a document with inline objects like images, then the techniques described in the MHTML standard should be used. `Content-Disposition` can be used, but should be ignored by mailers which understand `Content-Type: Multipart/related`.

1.8.3 Sending HTML in e-mail

MIME was planned in order to allow messages to contain pictures, formatted text, sound, etc. In practice, MIME has mostly been used to send such information as attachments. The success of the World Wide Web (WWW) means that the HTML (Hypertext Markup Language) has become a very successful format for documents with formatted text and inline graphics. There are two ways of sending HTML in e-mail:

(1) Use the message/external-body MIME body part, which means that the e-mail message only contains the URL (Uniform Resource Locator) of the actual message, and the recipient mailer will then use this URL to retreieve the message from the net just like viewing an ordinary web page.

(2) Send the HTML text within the message, using the Content-Type Text/HTML.

The HTML format often uses more than one file to convey a message. For example, graphics, frames and applets are usually stored in separate files, which are combined by the web browser when displaying the document. Thus, there is a need to send HTML as a set of related body parts which together form the document. This is done using the `Content-Type: Multipart/related`. The main message will then reference the other parts either through their Content-IDs or through their URLs. If URLs are used, the other body parts are given a heading `Content-Location` which indicates the URL of this body part.

The MHTML standard is not only useful for sending HTML in e-mail. It can also be used for archiving of web pages. By archiving web pages in the MHTML format, all information in a web page can be archived in a single file, instead of separare archiving of the HTML text and the in-line images and other in-line data.

1.9 SMTP – The Protocol for Message Transport

The Internet protocol for message transport is SMTP (except for the transmission from the final MTA to the recipient UA, where other protocols are usually used). SMTP is a protocol for the transmission of a message from the original UA to the first MTA and from one MTA to another MTA. SMTP is thus a protocol for transmission one step on the way from originator to recipient. However, some information, such as the e-mail address of the sender

is usually conveyed from one SMTP step to the next SMTP step.

Since SMTP is used to control the transmission, it specifies most of the envelope information. Some envelope information is however specified in the message heading.

SMTP is a session-oriented protocol. By that is meant that a connection is established between two agents, and then a series of interactions takes place between the two agents. Here is an example of a simple SMTP session:

Sending agent command	*Responding agent response*
Opens a connection to port 25 of the receiving host	Listens for connections on port 25, acknowledges new connections
`HELO dsv.su.se`	`250 nexor.co.uk`
`MAIL FROM: <jpalme@dsv.su.se>`	`250 OK`
`RCPT TO: <j.onions@nexor.co.uk>`	`250 OK`
`RCPT TO: <seb@nexor.co.uk>`	`250 OK`
`DATA`	`354 Start mail input`
... the lines of text in the message ...	
`.`	`250 OK`
`QUIT`	`221 nexor.co.uk service closing`

An SMTP session can include the transmission of more than one message. In that case, a new MAIL FROM command comes at the place of the QUIT command in the figure above.

The basic SMTP commands are:

"HELO" opens an SMTP session.

"MAIL FROM:" starts the sending of a new message by specifying the sender. Its value is an e-mail address enclosed in angle brackets.

"RCPT TO:" is repeated once for every recipient. In the original SMTP protocol, an acknowledgement from the server (the 250 response code) was required after every recipient, before the next recipient address could be sent. SMTP has later on been extended with an optional facility to send several recipients in sequence.

"QUIT" indicates the end of an SMTP session, a server receiving a QUIT command should close the connection.

"DATA" signifies the start of the body of the message. The server response "354" indicates that the server expects more. After DATA, the body is sent. According to the original SMTP standard, the body must be a number of text lines, and finished by a line containing a single period. Line breaks in SMTP must always be a carriage return character plus a linefeed in sequence (CRLF) and the end of the body is thus signified by CRLF, a single period, and a second CRLF. In order to allow the sending of messages containing lines with only a single period, all lines which start with a period have an extra period sent at the beginning of the line.

Thus, the following text:	**Is in SMTP sent as:**

`The next sentence will start with a period:`
`.line starting with a period`
`The next sentence will contain a single period`
`.`
	`The next sentence will start with a period`
	`..line starting with a period`
	`The next sentence will contain a single period`
	`..`
	`.`

In addition to the HELO, MAIL FROM:, RCPT TO:, DATA and QUIT command, the

original SMTP protocol contained some additional commands. Most of these are either not used or not used very much today. Examples of these commands are:

VRFY to ask the server to verify that it has a mailbox with a certain e-mail address. A server may also give a positive response if it is willing to deliver a message to this recipient, even though the recipient mailbox is not on the same host as the server. The parameter to VRFY can be an abbreviated name, the response should be the full e-mail address, if the abbreviation is non-ambiguous.

TURN is a command which within a single SMTP session switches the roles between sender and recipient, so that the agent which started the SMTP session can receive mail from the MTA it connected to. TURN is useful if connection times are long, such as for dial-up modem connections. The increasing use of fixed lines has reduced the needs for the TURN command.

The most common digits in the SMTP reply codes are:

1.9.1.1.1.1 First digit:

1 Positive preliminary (not used in SMTP)
2 Success
3 Ready but requires additional info
4 Transient failure
5 Permanent negative

1.9.1.1.1.2 Second digit:

0 Syntax (error)
1 Information requested in reply
2 Transport service problem
5 Application-specific problem

1.9.1.1.1.3 Examples of reply codes to the MAIL FROM command:

250 Originator accepted
452 Out of local storage
500 Command syntax error

A number of proposals for extensions to SMTP have been developed. These extensions allow the sending of delivery-report requests, binary and 8-bit data, and an SMTP sender can check if a very large message can be received before sending it. These extensions are only to be used by extended SMTP servers, so there is also a protocol for two SMTP servers to query each other's capabilities at the start of an SMTP session. This protocol thus provides an extension mechanism to SMTP and is called ESTMP (Extended Simple Mail Transfer Protocol). The method of protocol extension used by ESTMP is that the server gives the client a list of which ESTMP extensions it supports, and the client must then only use those extensions which the server says that it supports. This method has many advantages compared to the method of indicating protocol level (like HTTP version 0.9 or 1.0 or 1.1) because a server can choose to provide certain extensions without having to support other less useful extensions.

In plain ESTMP, a session is started by the client sending the HELO command. A client which supports ESTMP send the EHLO command instead of the HELO command. The EHLO command only indicates that the client understands ESTMP, not that the client is capable of handling any particular ESMTP extension.

The table below lists the most important ESMTP extensions:

Service extension	Keyword	Parameters	Verb	RFC
Send	SEND	none	SEND	821, 1869
Send or Mail	SOML	none	SOML	821, 1869
Send and Mail	SAML	none	SAML	821, 1869
Expand	EXPN	none	EXPN	821, 1869
Help	HELP	none	HELP	821, 1869
Turn	TURN	none	TURN	821, 1869
Pipelining	PIPELINING	none	none	1854
Message size declaration	SIZE	adds optional parameter size-value ::= 1*20DIGIT no of octets	none	1870
Checkpoint/ restart	CHECKPOINT	adds optional parameter TRANSID to MAIL FROM command	none	1845
Large and binary MIME messages	CHUNKING	BDAT is used instead of DATA, and takes as parameter packet length and last packet indication	BDAT	1830
8bit-MIMEtransport	8BITMIME	adds optional parameter BODY to MAIL FROM, values 7BIT and 8BITMIME	none	1652
Delivery Status Notification Extension	DSN	adds optional parameters NOTIFY and ORCPT to RCPT command and RET and ENVID to the MAIL command	none	1891, 1892, 1894

Here are three different examples of ESTMP capability negotiations:

(1)
Only mandatory SMTP commands provided

```
S: <wait for connection on TCP port 25>
C: <open connection to server>
S: 220 dbc.mtview.ca.us SMTP service ready
C: EHLO ymir.claremont.edu
S: 250 dbc.mtview.ca.us says hello
```

(2)

```
S: <wait for connection on TCP port 25>
C: <open connection to server>
S: 220 dbc.mtview.ca.us SMTP service ready
C: EHLO ymir.claremont.edu
S: 250-dbc.mtview.ca.us says hello
```

Basic optional services: EXPN, HELP;

```
S: 250-EXPN
S: 250-HELP
```

Standard service extension: 8BITMIME;

```
S: 250-8BITMIME
```

Unregistered services: XONE and XVBR

```
S: 250-XONE
S: 250 XVRB
```

(3)
ESTMP not supported

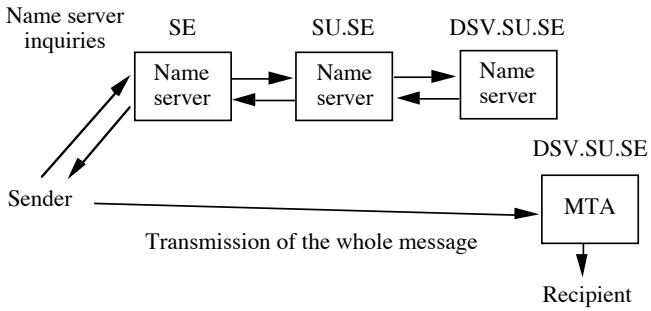
```
S: wait for connection on TCP port 25>
C: <open connection to server>
S: 220 dbc.mtview.ca.us SMTP service ready
C: EHLO ymir.claremont.edu
S: 500 Command not recognized: EHLO
```

1.9.2 SMTP command pipelining

SMTP often requires many interactions back and forward between client and server. For example, when a message is to be sent to many recipients, the client is expected to send a RCPT TO for each recipient and wait for the response from the server before sending the next RCPT TO. This will make the protocol slow, since interactions across large network distances often cause a delay of one or more seconds.

In practice, many SMTP clients ignore this rule, and send everything without waiting for reply codes, and then gets all the reply codes asynchronously. This is not correct, but usually works, since TCP will provide storage on the net of the data sent in advance. An ESMTP extension specified in RFC 1854 allows a server to indicate that it is capable ot such pipelining.

1.9.3 Use of domain addresses for routing



Name server look-up followed by direct transmission.

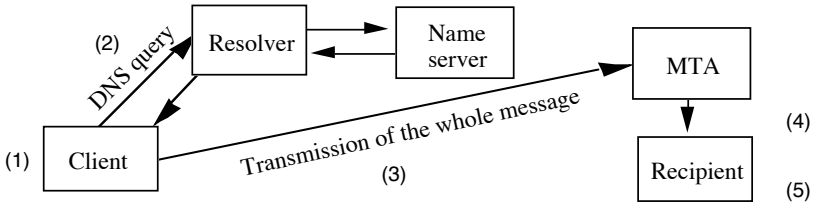
The figure above shows how a series of name servers that know host addresses in domains successively closer to the recipient can be used to find the network address of the recipient MTA host. The sender can then transmit directly to the recipient MTA. Of course, the name servers for SU.SE and DSV.SU.SE might be combined into one name server. And the name server for SE can cache names and addresses, so that after a second inquiry for the address of DSV.SU.SE, the SE name server can answer directly without forwarding the query to SU.SE. The technique of keeping names passing through a name server is called *caching*. Cached addresses should only be kept for a limited time, since they may otherwise become out-of-date.

The technique described in the figure above, where each successive name server connects to another name server, is called *chaining*. Another also commonly used technique is called *referral*. With referral, the searcher will successively connect to a series of name servers until the right one is found.

1.9.4 Routing and Use of Name Servers in Internet

The methods for routing mail messages using name servers in Internet is described in RFC

974 [18], RFC 1101 [23], RFC 1123 [20], and RFC 1348 [24].



Use of name servers for Internet mail routing.

E-mail handling in the Internet is usually done in the following stages. (The numbers refers to numbers in the figure above.)

(1) The originator edits and submits his message for mailing.

(2) For each recipient, the name server facility is used to find the IP-address of the host which is most suitable for delivering mail to the recipient.

(3) The mail is then forwarded to the hosts serving each recipient. The SMTP protocol is used for this.

(4) The host described in (3) can be the host closest to the recipient or an intermediate host which forwards the message, for example, a gateway to another network with a different mail standard. If it is an intermediate host, this host will then use SMTP or some other protocol (like X.400 P1) to forward the message to the next host in a chain leading to the recipient. If the message is sent to a distribution list, the host handling the list will expand the list and forward the message to the members of the list. Often, incoming messages from the Internet to an Intranet are first routed to a firewall process, which will check the messages for viruses, before they are re-routed internally.

(5) Finally, the recipient reads the mail. The mail system of the recipient can use the machine-readable information in the message heading to aid the user in providing facilities like:

- (a) Finding the message to which the current message is a reply.
- (b) Finding messages from a certain sender, or messages which arrived via a certain distribution list.
- (c) Finding messages written between certain dates.

Name servers for routing in the internet are called DNS servers. A DNS server takes as input an Internet domain address, such as EIES2.NJIT.EDU. There are many DNS name servers, all responsible for only part of the DNS tree. This means that sometimes the first name server contacted by a resolver may not have the information requested. The information can then be found by using either *chaining* or *referral*. The Internet DNS allows both chaining and referral. Every DNS server must support referral. Support for chaining is optional.

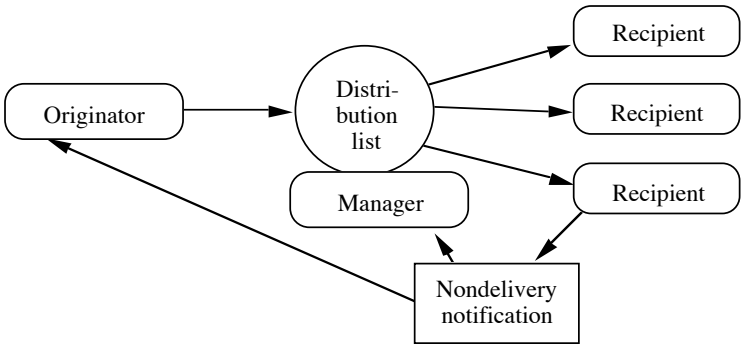
1.9.5 Delivery Status Notifications

A good electronic mail system should always (unless the sender explicitly relinquishes this requirement) inform the sender if a message cannot be delivered. But sometimes a fault, such as a disk crash, can cause messages to disappear without any notification to the sender (such situations are often called *black holes*). However, if the sender has requested a delivery

notification, the fact that it has not arrived within a reasonable time can indicate to the sender that the message has been lost, even when no nondelivery notification appears.

It is, of course, advantageous to the sender if his mail software automatically recognizes incoming delivery and nondelivery notifications and handles them in suitable ways. The sender may prefer not to be informed every time such a notification arrives, but to store them so that later the sender can ask the system to provide a report about which recipients have received their messages. Notifications usually indicate that a message has reached the mailbox of the recipient, but some systems also provide notifications when a message has been read by the recipient, or, rather, when it has been shown on the screen of the recipient.

Nondelivery notifications are often difficult to understand if you are not an expert on electronic mail protocols. Getting a nondelivery notification from some recipient to whom you never sent any message can be especially confusing. What happens is that you send a message to a mailing list, and there is a delivery problem in delivering the message to one of the members of this list. For large mailing lists, such delivery reports should, of course be sent to the manager of the list, not to the originator of the message. However, it is not uncommon for mailers to mistakenly send the nondelivery notifications to the originator.



Sending error reports to mailing list manager or to the originator

The Internet mail delivery notification functionally is called Delivery Status Notifications (DSNs), and it is specified in four RFCs:

RFC 1891: SMTP service extension, 31 pages

RFC 1892: Multipart/report content-type, 4 pages

RFC 1893: Enhanced status codes, 15 pages

RFC 1894: Delivery Status Notification format, 31 pages

Requests for delivery status notifications is sent via SMTP, using optional parameters to the RCPT TO and MAIL FROM commands:

SMTP command	Optional parameter	Description
`RCPT TO`	`NOTIFY`	Values: NEVER = do not send any DSNs. SUCCESS = send a DSN if the message was successfully delivered to the recipient mailbox FAILURE = send a DSN if the message could not be delivered to the recipient mailbox DELAY = indicate willingness to accept notifications if delivery is delayed but may succeed later on
`RCPT TO`	`ORCPT`	Original sender-specified recipient address (needed to allow recpient of notifications to correlate notifications with original recipients, since some gateways will rewrite the recipient e-mail addresses before delivery)

SMTP command	Optional parameter	Description
MAIL FROM	RET	Values: FULL = return full text of the message with the DSN HDRS = return only headers of the message with the DSN If neither FULL nor HDRS is indicated, this means that no return of either headers or body is requested.
MAIL FROM	ENVID	Transaction ID to be returned with notification, so that the sender can find which message sending caused the failture. (Message-ID cannot be used for this, since sometimes the same message is sent more than once with the same Message-ID).

While the requests for DSNs are sent via SMTP, the DSNs themselves are specially formatted MIME messages. A DSN is sent as a MIME message with the Content-Type Multipart/Report. A Multipart/Report consists of two mandatory and one optional part:

Part 1 (mandatory) is a human readable message explaining in natural language the cause of the error. This part is mainly intended for recipients whose e-mail clients do not recognize the special format of Part 2 of a DSN.

Part 2 (mandatory) contains a machine parsable account of the reported event, with a special MIME type called Message/Delivery-status.

Part 3 (optional) contains the original message either full or only its headers.

Here is an example of a complete delivery status report:

```
Date: Thu, 7 Jul 1994 17:16:05 -0400
From: Mail Delivery Subsystem <MAILER-DAEMON@CS.UTK.EDU>
Message-Id: <199407072116.RAA14128@CS.UTK.EDU>
Subject: Returned mail: Cannot send message for 5 days
To: <owner-info-mime@cs.utk.edu>
MIME-Version: 1.0
Content-Type: multipart/report; report-type=delivery-status;
      boundary="RAA14128.773615765/CS.UTK.EDU"

--RAA14128.773615765/CS.UTK.EDU Part 1, in “human-readable” (?)
format:

The original message was received at Sat, 2 Jul 1994 17:10:28
-0400
from root@localhost

   ----- The following addresses had delivery problems -----
<louisl@larry.slip.umd.edu>  (unrecoverable error)

   ----- Transcript of session follows -----
<louisl@larry.slip.umd.edu>... Deferred: Connection timed out
      with larry.slip.umd.edu.
Message could not be delivered for 5 days
Message will be deleted from queue

--RAA14128.773615765/CS.UTK.EDU Part 2, in machine-parsable
format:
content-type: message/delivery-status

Reporting-MTA: dns; cs.utk.edu

Original-Recipient: rfc822;louisl@larry.slip.umd.edu
Final-Recipient: rfc822;louisl@larry.slip.umd.edu
Action: failed
Status: 4.0.0
```

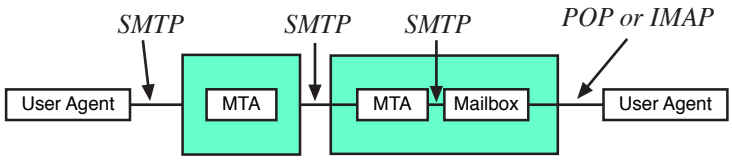
```
Diagnostic-Code: smtp; 426 connection timed out
Last-Attempt-Date: Thu, 7 Jul 1994 17:15:49 -0400

--RAA14128.773615765/CS.UTK.EDU
content-type: message/rfc822    Part 3, return of original
message:

[original message goes here]
--RAA14128.773615765/CS.UTK.EDU--
```

Here are some of the fields which can occur in a Delivery Status Report:

`per-message-fields =`	Fields which apply to the whole message.
`[ original-envelope-id-field CRLF ]`	Envelope identifier from request.
`reporting-mta-field CRLF`	MTA which attempted to perform the delivery or relay.
`[ dsn-gateway-field CRLF ]`	Name of gateway which transformed foreign delivery report.
`[ received-from-mta-field CRLF ]`	MTA from which the message was received.
`[ arrival-date-field CRLF ]`	Arrival date to reporting MTA.
`*( extension-field CRLF )`	
`per-recipient-fields =`	Fields which apply to only one recipient at a time.
`[ original-recipient-field CRLF ]`	Original recipient when sent.
`final-recipient-field CRLF`	Final recipient to whom delivery status is reported.
`action-field CRLF`	failed, delyaed, delivered, relayed (to non-DSA environment), expanded.
`status-field CRLF`	Status code (RFC 1893) (DIGIT "." 1*DIGIT "." 1*3DIGIT
`[ remote-mta-field CRLF ]`	Name of MTA which reported to reporting MTA.
`[ diagnostic-code-field CRLF ]`	Sometimes less preciste diagnostic code from remote MTA.
`[ last-attempt-date-field CRLF ]`	Time of last delivery attempt.
`[ final-log-id-field CRLF ]`	Log entry in final MTA logs.
`[ will-retry-until-field CRLF ]`	Time when delivery attempts will stop.
`*( extension-field CRLF )`	

1.10 Message delivery

When a user runs an e-mail client package on his personal computer, this client needs a protocol to talk to a server, corresponding to the P3 and P7 protocols in X.400. The model behind these protocols, like in X.400, is that mail messages are stored in a server, to be downloaded to the e-mail client at request of the user, as is shown by this picture:



Model behind the POP and IMAP protocols.

In the Internet, the two most important such protocols are:

- Post Office Protocol (POP) [9, 12], a protocol for fast downloading of mail to client software, where the client stores and handles the mail in the personal computer, corresponding to P3 in X.400.
- Interactive Mail Access Protocol (IMAP) [8], a protocol for cases where the user wants to store his messages in the server, and wants to be able to manipulate this storage from client software on his personal computer. IMAP is a more complex protocol than POP.

Commands in POP and IMAP are textual strings, just like commands in SMTP. Here is a list of the most important commands in POP:

USER	Client identifies mailbox to be downloaded
PASS	Password
STAT	Get number of messages and size of mailbox
LIST N	Return size of message N
LAST	Get highest message number accessed
RETR N	Retrieve a full message
TOP N M	Retrieve only headers and the first N lines
DELE N	Delete message
QUIT	Release service
NOOP	See if POP server is functioning
RPOP	Insecure authentication

IMAP is a more sophisticated protocol than POP. In IMAP, a server can send messages to the client without a request from the client, and several transactions between server and client can go on in parallel. This can be used to reduce the wait time for the users. Each message in an IMAP mailbox has a set of properties, which can be retrieved one or more than one at a time. Examples of properties are a seen flag and a deleted flag on the message. In IMAP, to delete a message you first set the delete flag on the message, and then perform the expunge command. IMAP also has a capability for searching for messages in the mailbox stored in the server.

1.11 Mailing lists

1.11.1 Expansion of nested mailing lists

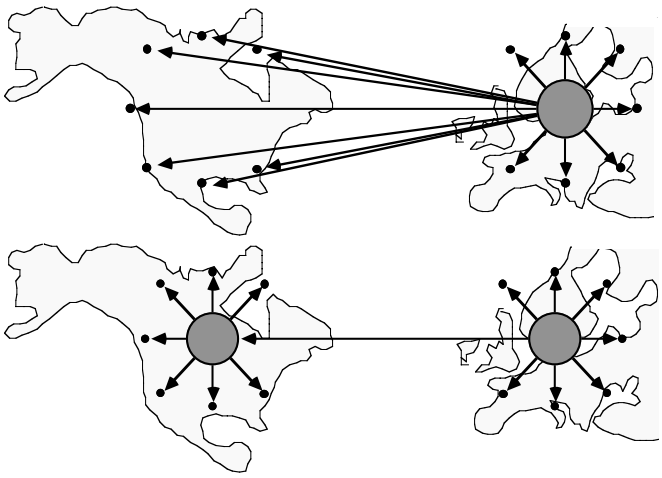
Nested mailing lists occur when one list is a member of another list. If, for example, list B is a member of list A, then a message sent to list A will be distributed to all members of both list A and B. A message sent directly to list B, however, will not reach the members of list A, unless A also is a member of B.

There are two techniques for handling mailing lists:

- *Sender UA expansion.* The mailbox software (user agent software) of the sender finds the list of the members of the list and sends it directly to them. If the lists are nested, the sender UA will successively and recursively find the lists of members of the sublists, all the way to the final recipient. With this method, the message will thus be converted to an ordinary multirecipient list. The message will also have ordinary users, and not lists, as recipients, before the message leaves the sender UA .

- *Expansion at the list location.* The sender's UA sends the list to the list-expansion agent or to the domain which is responsible for the list. The list is then expanded at this location. Expansion means the replacement, on the envelope (not in the message heading) of the name of the list with the names of the members of the list. With nested lists, the message is then forwarded to the domains of the sublists, which perform the secondary expansion, and so on if there are sublists to the sublists.

Consider a person who is a member two lists. With the second method, this person will probably receive two copies of the same message. With the first method, such duplicates can be eliminated during the expansion. In spite of this, expansion at the list location (the second method) is most common. With two nested lists, one for the European and one for the American members, the message need only be sent to one single recipient when crossing the Atlantic. If the whole list is expanded by the sending UA, at least 100 recipient have to be listed when crossing the Atlantic, which will make the transfer more expensive. See the figure below.



Advantage of using nested mailing lists (bottom) vs. nonnested (top).

Other advantages with expansion at the list is that it is easier, to support distributed control of the membership of a group—each sublist can have its own management. This can, of course, be a disadvantage when central control of the membership is required.

1.11.2 Loop control for nested mailing lists

If two lists are directly or indirectly members of each other, there is a risk that the same message will be looped back and forward indefinitely between the lists. There are several different techniques for avoiding this:

(1) Full expansion by the originating UA.

(2a) A trace list of all the mailing lists passed is put on the envelope of the message. A mailing list can then refuse to accept, incoming messages, that have the name of the mailing list itself as part of the trace list on the envelope of the message.

(2b) Using a variant of method (2a), each mailing list will instead refuse to send a message to another mailing list which is included in the trace list of the message.

(3) The registration system for mailing lists is designed in such a way that no list will ever be a member, directly or indirectly, of itself.

(4a) Each mailing list stores the message-IDs of all messages passing through the list. When the same message returns once more to the list, the list checks the message-ID of the message and stops the loop if a message with the same ID has already passed through the list. (Message-ID is also known under the term Message-ID.)

(4b) This is a variant of method (4a), where a checksum of the content of the message is used instead of the message-ID.

(4c) Another variant of method (4a) is to only stop resending a message with the same Message-ID to the same outgoing recipient.

An advantage of method (2b) is that it saves some unnecessary transmission. However, method (2b) is not as reliable as method (2a), because the same electronic mail address can have different forms, and therefore the comparison used in method (2b) may not work. It is easier for a list expander to recognize a name which it itself puts on an envelope than a name which some other list expander puts on an envelope.

A problem with method (2a) and (2b) is that it is not enough to just but the name of the MTA in the received header, since a message can legitimately pass the same MTA more than once. Some systems, because of this, only stop a message when the MTA name occurs 3 or 12 times in the message header. This will not stop a loop completely, but at least avoid the loop to continue infinitely.

Comparing methods (4a) and (4b), method (4b), use of a checksum, has the advantage that it caters for systems where the message-ID is corrupted (something which is not unusual), but has as a disadvantage that two different messages may accidentally get the same checksum. A further problem with the message-ID is that two messages with the same message-ID may not always be identical. Finally, some mail systems do not generate globally unique message-IDs on the messages they produce.

Method (4a) is often used by computer conferencing systems, sometimes combined with the other methods. For example, Usenet News mainly uses method (4a).

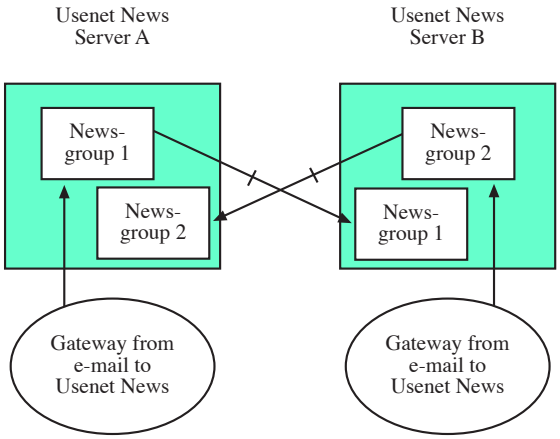
Programs that send messages through gateways from Internet e-mail to Usenet News will assign Message-IDs to messages lacking such IDs. This can cause the same message to get different Message-IDs by different gateways.

The Listserv software has very powerful methods for avoiding loops, primarily based on method (4b).

In practice, many existing mailing lists have no loop control mechanism except that the people who manage the list manually try to avoid producing loops.

Usenet News uses Message-IDs as its main loop control mechanism. This is, in Usenet

News, implemented in a way which will sometimes cause multi-group messages to only appear in some of the groups, to which it was sent. The figure below explains how this can happen:



The original message is sent by e-mail to two e-mail mailing lists,which are both gatewayed from e-mail to Usenet News through two gateways. One of the gateways enters one of the mailing lists to Newsgroup 1 through Usenet News server A. The other gateway ensters the other mailing list to Newsgroup 2 through Usenet News server B.

When the Newsgroup 1 version of this message is to be moved from news server A to news server B, the loop control in news server B will refuce to accept the message, since it already has the message in Newsgroup 2. This means that subscribers to Newsgroup 1, but not Newsgroup 2, in Usenet News server B, will incorrectly never see this message. In the same way, subscribers to only Newsgroup 2 in News server A will never see the message.

I have suggested that the Usenet News standards should be modified to remove this problem. But the experts in IETF say that this is not possible, we will have to live with this problem. This is not the first time when the technical experts in IETF have refused user-friendly changes to standards.

1.11.3 Management of large mailing lists

Management of large mailing lists is not easy. The manager will every day receive nondelivery notifications for messages from the list which cannot be delivered, and requests for addition and removal of members from the list. It is sometimes difficult to identify the item in the list of members which such a notification or request refers to. Sometimes, they are actually members of a sub-mailing list. The problems of managing large mailing lists is more fully discussed in [3].

1.11.4 How You Become a Member of a Mailing list

1.11.4.1 The LISTSERV Way

Suppose you want to become a member of a mailing list, whose e-mail address is listname@host.bit.net. You then write an e-mail message addressed to listserv@host.bit.net and write in this message a single text line with the text:

```
SUB listname Your Own Name
```

For information about other commands you can give, such as how to unsubscribe from a list, download archived messages from the list, etc., write a message to the same recipient, listserv@host.bit.net, with the single word HELP in the text of your message. For more information about Listserv.

1.11.4.2 The -Request Way

Suppose you want to become a member of a mailing list, whose e-mail address is listname@host.bit.net. You then write a message to listname-request@host.bit.net and write in the text of the message something like “Please add me as a member of this mailing list.” The e-mail address with the name of a mailing list with “-request” added to it can also be used for other communication with the list manager.

1.11.4.3 Subscription through web pages

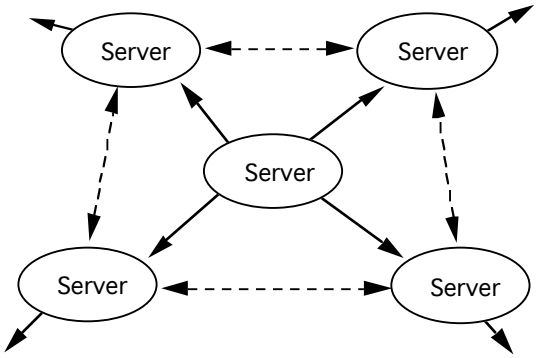
A third method is to subscribe to mailing lists through web pages. However, there is usually no very secure identification of a person who accesses a web page. Because of this, a convention has developed that if a person subscribes to a mailing list through a web page, then the list server will first send an e-mail to the user, asking him/her to confirm the subscription. Not until this confirmation is received, will the subscription be entered.

1.12 Usenet News

Usenet News is a distributed asynchronous forum system. Forums in Usenet News are called newsgroups, and messages are called articles.

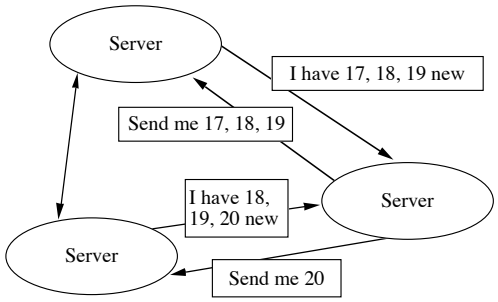
1.12.1 Distribution of news

The basic principle of Usenet News is that a local server handles most of the functionality. Usenet News standardizes two variants of the NNTP protocols: One for communication between adjacent servers, one for communication between a client and a server. Each server can download as much as it wants of what is available on any of the adjacent servers. Loop control is handled both by a trace list and a list of the Message-IDs of received messages stored by each server, so that the server can reject the same message coming back again. The procedure for distribution of news can be compared to pouring water onto a flat surface; the water flows out in all directions as shown in the figure below.



“Pouring water” principle of Usenet News distribution.

The figure below shows how new articles are forwarded from server to server in Usenet News. A server tells its adjacent servers which items it offers, the server requests those it has not already got via another route.



This figure shows how new articles are forwarded from server to server in Usenet News.

Information about a user, such as how much this user has seen, is stored in the client. The server need not even know which users are using it. There are many different user-interface softwares for Usenet News, Some of them, of course, do not provide all the available functions.

1.12.2 Cancel and Supersedes

In addition, Usenet News provides an interesting functionality which restricts communication to only those members of a newsgroup who work in the same organization or live in the same area or country. This functionality, however, is not used very much, and its existence is controversial, since it means that different users will get different views of the same newsgroup.

Usenet news has a *cancel* command, which can delete messages already sent out. Only the author of the cancelled message and the local newsserver administrator is allowed to cancel a message. Since, however, it is very easy to fake your identity, this command poses an obvious security risk, and the command is known to have been used to cancel messages for political reasons. The command is also used (not quite appropriate) by cancelbots, robots (= automatic programs) which cancel obvious spams by identifying messages with the same content sent to many disparate newsgroups. Usenet news also often has a Supersedes header field, which refers from a new message to an old message. This header usually cancels the old message.

There is also a *supersedes* header, which is used when a new message is sent to replace the previous version of this article. Supersedes is similar to cancel in that it causes a real deletion of the message being replaced. This is different from, for example, the obsoletes header of X.400, which only marks a new message as a replacement, but is not meant to cause the previous version to be physically deleted. Obsoltes thus is similar to the In-reply-To and References headers in e-mail.

The most important restriction of Usenet News is that closed groups are not well supported. The only kind of closed groups which are common in Usenet News are groups which are restricted to one or a selected set of news servers. Such groups will then be open to anyone with an account in these news servers, but closed to everyone else.

1.12.3 The Network News Transfer Protocol (NNTP)

The Usenet News protocol is called network news transfer protocol (NNTP) and is specified in RFC 977 [46]. The standard for the format of Usenet News articles is specified in RFC 1036 [19].

The table below lists the most common NNTP commands:

`article [<Message-ID> | <Number>]`	Return text of designated article. If no parameter is given, the next article is returned. The current article pointer is put at the fetched article.
`body [<Message-ID>| <Number>]`	As article, but only returns body
`group <newsgroup>`	Go to the designated newsgroup
`head [<Message-ID> | <Number>]`	As article, but only returns head
`help`	Lists available commands
`ihave <messageID>`	Informs the server of an available article. The server can then ask for the article or refuse it.
`last`	Sets current article pointer to last message available, return the number and Message-ID.
`list [active | newsgroups | distributions | schema]`	Returns a list of valid newsgroups in the format: *group last first*
`newgroups <yymmdd hhmmss> ["GMT"] [<distributions>]`	List newgroups created since a certain datetime. "distributions" can be e.g. *alt* to only get newsgroups in the *alt* category.
`newnews <newsgroups> <yymmdd hhmmss> ["GMT"] [<distributions>]`	List Message-ID of articles posted to one or more newsgroups after a specific time. *newsgroups* can be. e.g. net.*.unix to match more than one newsgroups. *distributions* checks for articles which also has this other newsgroup as recipient.
`next`	Current article pointer is advanced. Returns number and Message-ID of current article.
`post`	Submit a new article from a client.
`slave`	Tells the server that this is not a user client, it is a slave server. (May give priority treatment.)
`stat [<Message-ID> | <Number>]`	As article, but only returns Message-ID. Used to set the current article pointer.

Note that the same protocol, NNTP, is used for communication both between a client and a server, and between two servers, but that sometimes different commands are used. Thus, when a user client submits a new message, the `post` command is used, but when a server

sends a new message to another server, it usually uses `ihave` to give this information, and the receiving server then requests the message, if it has not already received it from another server.

Most of the message headers are the same in Usenet News and in e-mail, but there are some differences as shown by this table:

`Newsgroups:`	Comma-separated list of newsgroups to which this article belongs. Example of newsgroup format: alt.sex.fetisches.feet. *Should never occur in e-mail*. Use "Posted-To:" instead!
`Subject:`	Add four characters "`Re.` " for replies. Do not change subject in replies.
`Message-ID:`	Mandatory in Network News, and must be globally unique.
`Path:`	Path to reach the current system, e.g. abc.foo.net!xyz!localhost. E-mail path format also permitted. Compare to `Received:` and `Return-Path` in e-mail.
`Reply-To:`	In news: Where replies to the author should be sent. In e-mail: Ambiguous.
`Followup-To:`	Where replies to newsgroup(s) should be sent.
`Expires:`	Suggested expiration date.
`References:`	Message-ID-s of previous areticles in the same thread. Should always contain first and last article in thread. Compare to e-mail: Usually only immediately preceeding messages..
`Control:`	Not used in e-mail. Communication with servers. Body or subject contains command. Subject begins with "cmsg".
	`cancel` Delete physically a previously sent article.
	`ihave` Host telling another host of available new articles.
	`sendme` Host asking for articles from another host.
	`newgroup` Name of new group, plus optional word moderated.
	`rmgroup` Remove a newsgroup. Requires approved.
	`sendsys` Send the sys file, listing neighbours and newsgroups to be sent to each neighbour.
	`version` Version of software wanted in reply.
	`checkgroups` List of newsgroups and descriptions, used to check if list is correct.
`Distribution:`	Not used in e-mail. Limits distribution to certain geographical/organizational area. Example: Distribution: se, no.
`Organization:`	Of sender.
`Keywords:`	For filtering.
`Summary:`	Brief summary.
`Approved:`	Required for message to moderated group. Added by the moderator, contains his e-mail address. Also required for certain control messages.
`Lines:`	Count of lines of the message body.
`Xref:`	Numbers of this message in other newsgroups. Only for local usage in one server. Example: Xref: swnet.risk:456 swnet.sunet:897

There is a problem with the `newsgroups` header in a message sent via both Usenet News and e-mail. Different systems use this header in two different ways, in the mail version of such messages:

(a) To indicate that this message has also been sent via Usenet news to the indicated newsgroups.

(b) To indicate that this is a personal reply, sent only via e-mail, to a message posted on the indicated newsgroups.

Because of this problem, it is better to use the `Posted-To` header in e-mail to indicate that a message has also been sent to certain newsgroups, and e-mail recipients should ignore any `newsgroups` heading in an e-mail message.

MIME is not used as much in Usenet News as in e-mail. Instead of BASE64, an older method named UUENCODING is often used in Usenet News to include binary attachments. Also, because of message size restrictions, large attachments are very often split into several messages in Usenet News. This also occurs in e-mail, but is more frequent in Usenet News, since some Usenet News servers try to save space by not accepting articles above a certain size limit. Both MIME and Usenet News have methods of indicating how a client can automatically combine parts into a complete message or attachments.

In addition to NNTP, also other protocols are sometimes used for communication between Usenet news servers.

1.12.4 News Control in Usenet News

An important functionality in all asycnchornous (not same time) message systems is the news control functionaltity. This functionality will help the user find which messages are new and not yet read by this user. Most message systems handle this functionality together with the storage of messages. This means that if messages are stored in a server, the server also knows for each user, what that user has read andn ot read. And if messages are stored in the user's personal computer, then news information is also stored there.

Usenet News, however, does this in a different way. Messages are usually stored in the server, but the server does not know which messages each individual user has read and not read. The information on what a user has read is stored in a very compact format in the user agent software. The most common format for this file is the one shown by this example:

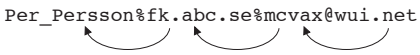
```
254-290
300-312
350-
```

The example above says that this user has unseen articles number 254-290, 300-312 and all articles numbered higher than 350. This format is very compact, because there are usually long sequences of articles which are read and unread. Storing a single bit for each message is then not optimal.

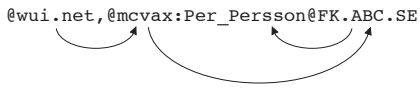
1.13 Relative addresses

Domain addresses are *absolute* addresses. An absolute address is the same address for a certain recipient, irrespective of where the message is sent from. Another type of address is called a *relative* address. A relative address indicates one or more relay stations on the route to the recipients. This would be roughly equivalent to indicating the postal mail address of a person as: “First to London, then from London by train to Nottingham, then by truck to the University, then by mailman from the University to the Computer Science department.”

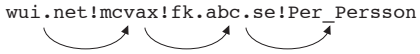
There are three commonly used formats for relative electronic mail addresses. All the three addresses in the figure below indicate the same relative route but in a different printed format



This format is sometimes used by gateways to the Internet, although it is not officially part of the Internet mail standards.



This format is sometimes used in the Internet mail standards.

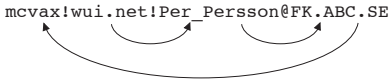


This format has been used much, and is still sometimes used, in Usenet News.

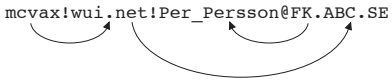
Combinations of several of these formats in one address may occur sometimes. For example, an address of the format:

```
MCVAX!FK.ABC.SE!Per_Persson@WUI
```

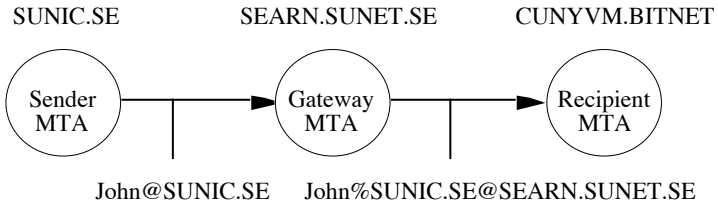
may according to some practices be interpreted as:



but may according to Usenet News practice be interpreted as:



The format using percent (%) signs is not part of the official Internet mail standards. This format will occur sometimes often when a message passes a gateway between two nets. The figure below shows how gateways can create relative addresses.



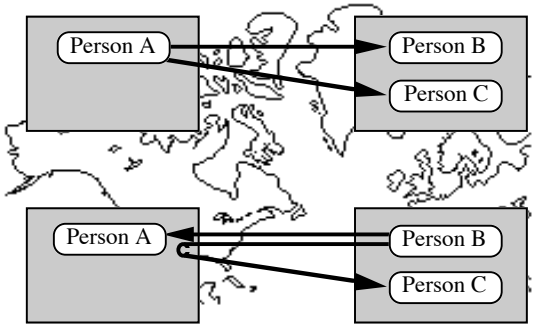
A message from a sender **John@sunic.se** passes a gateway from Internet to Bitnet. The address of the sender is given as **John@sunic.se** before the gateway, but the gateway translates this into **John%sunic.se@searn.sunet.se** when transmitting the message to Bitnet. Thus, the gateway takes the original sender address, replaces the @ with a %, and appends @ followed by the name of the gateway.

In Bitnet, when a reply is sent from the recipient to the sender, the reply is first sent to **John%sunic.se@searn.sunet.se**. Bitnet views this address as a user named **John%sunic.se** at a host named **searn.sunet.se**. Thus, from a Bitnet viewpoint, the whole Internet is in this case seen as local names in **searn.sunet.se**. When the reply reaches searn.sunet.se, this MTA will strip out **@searn.sunet.se** and look at the rest, **John%sunic.se**. It will translate this back to **John@sunic.se** and forward the reply to the original sender.

This example shows that neither the Internet nor the Bitnet will actually view this address as relative. In Bitnet, the address is an absolute address of a user named **John%sunic.se** in the host searn.sunet.se. Only the gateway itself recognizes the translation between % and @.

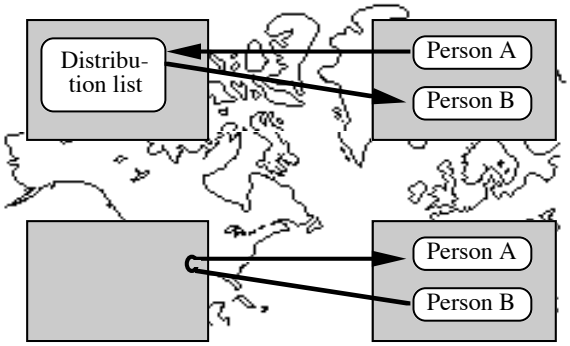
Thus, fundamentalists can proudly claim that “our messaging standard does not use any relative addressing” even though relative addresses are sometimes transported in their net.

There are many disadvantages of relative addresses. You cannot print your address on your business card in the same way for all recipients. You have to indicate a different address depending on to whom you are giving your address. There may also be problems with sending a return message, as shown by the figure below:



Suppose that a person A in America sends a message to two recipients B and C in Europe and that this message uses relative addressing via some American gateway. This means that when B gets the message, the relative address to C is given via this American gateway, so that if B sends a reply to C, this reply must be routed unnecessarily twice across the Atlantic. There are two disadvantages of this. Firstly, it means unnecessary costs and delays, as was shown in the example above. Secondly, the American net may refuse to transmit messages from a European sender to another European recipient, since the American net may have to pay for part of this unnecessary transmission.

A problem similar to that in Figure «#techniques».«#relativeinefficiency» can occur when distribution lists are used, as in Figure «#techniques».«#relative2inefficiency».



In this example, a person A in Europe sends a message to a distribution list in America. The letter reaches a recipient B in Europe via the list. If relative addressing is used, a reply from B to A may have to be transferred via a gateway in America.

Everyone agrees that relative addressing is bad. Why then does it crop up again and again? The reason is that electronic mail is handled by different electronic mail networks, connected via gateways. Since each net has limited knowledge of the internal structure of another net, it can often only address a recipient in another net via a gateway between the nets, and the inclusion of such gateways into addresses makes them relative. The increasing dominance of

Internet as a single universal network has however led to relative addressing occuring less and less often.

1.14 Instant Messaging

(Not yet ready)