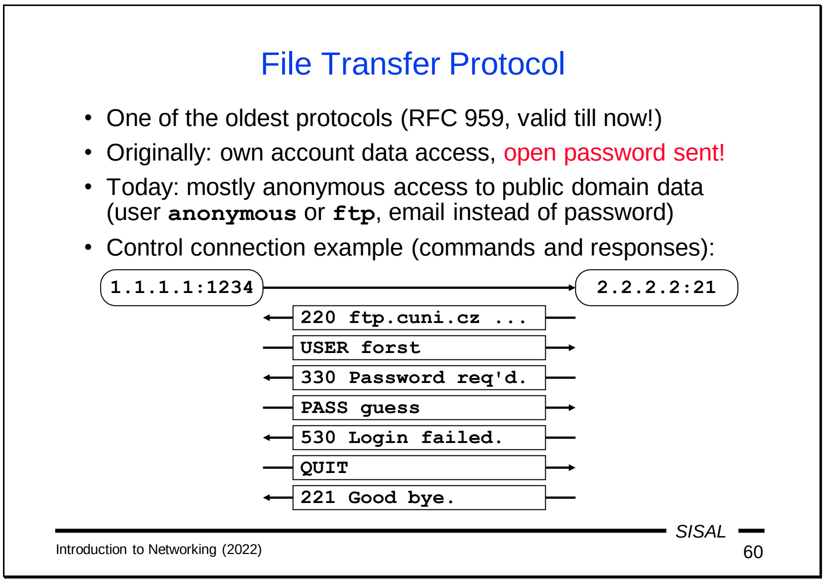
Slide 60



File Transfer Protocol

- One of the oldest protocols (RFC 959, valid till now!)
- Originally: own account data access, open password sent!
- Today: mostly anonymous access to public domain data (user **`anonymous`** or **`ftp`**, email instead of password)
- Control connection example (commands and responses):

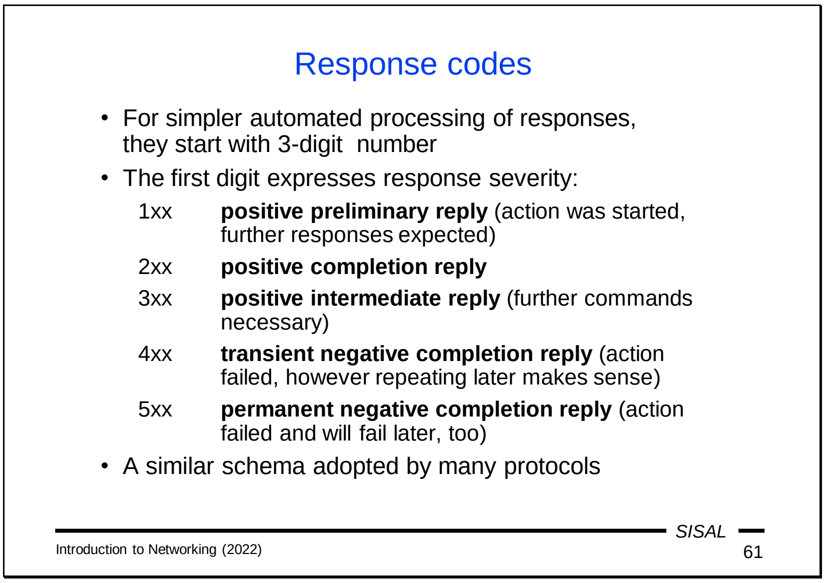
```
1.1.1.1:1234 ──────────────────────────────→ 2.2.2.2:21
          ←── 220 ftp.cuni.cz ...      ───
          ─── USER forst               ──→
          ←── 330 Password req'd.      ───
          ─── PASS guess               ──→
          ←── 530 Login failed.        ───
          ─── QUIT                     ──→
          ←── 221 Good bye.            ───
```

Introduction to Networking (2022) SISAL 60

The File Transfer Protocol (FTP) is one of the oldest protocols still in use today. Its original purpose was to allow a user account to transfer files from or to a remote computer. To log in in this way, it is necessary to authenticate the user, and in FTP, this is done by transmitting a password in the clear, which poses an obvious security risk. However, this approach was soon supplemented by an equally (and perhaps more) important but less risky variant, in which the user logs in as an anonymous user and instead of a password, a user provides an email address (for statistical purposes only). A user logged in in this way has access to freely available documents, programs, etc. In its time, FTP was an information source that was as important as the WWW is today. And some servers still use FTP archives without the user's knowledge: web browsers display a link to a resource available via FTP in the same way as a link to HTTP, and when the user clicks on such a link, the browser establishes an FTP connection and makes the result available to the user. A typical user does not have to know at all that the whole transmission took place by another protocol.

FTP is a text based protocol; the client establishes a so-called *control connection* to the server on port 21 and then sends command lines, while the server sends response lines over the same channel.

Slide 61



Response codes

- For simpler automated processing of responses, they start with 3-digit number
- The first digit expresses response severity:

  1xx **positive preliminary reply** (action was started, further responses expected)

  2xx **positive completion reply**

  3xx **positive intermediate reply** (further commands necessary)

  4xx **transient negative completion reply** (action failed, however repeating later makes sense)

  5xx **permanent negative completion reply** (action failed and will fail later, too)
- A similar schema adopted by many protocols

SISAL

Introduction to Networking (2022) 61

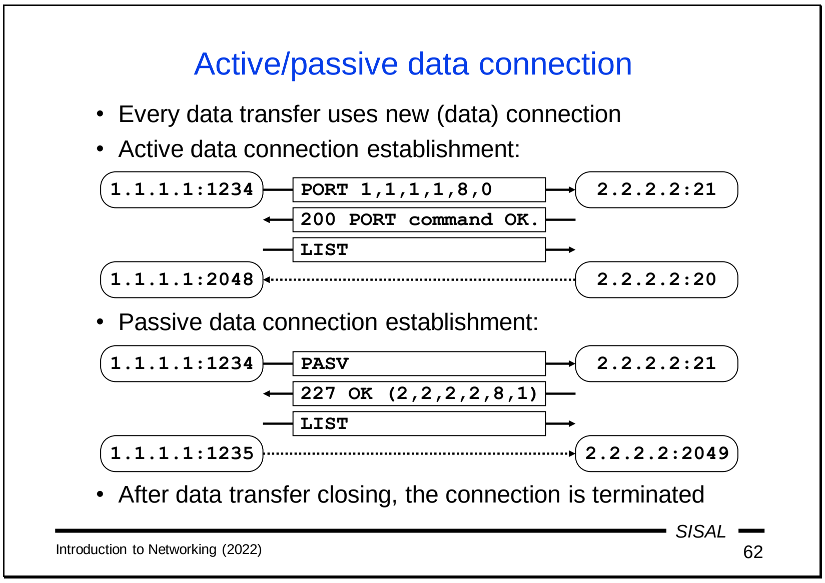
FTP was one of the first protocols designed to identify responses with a three-digit code, where the first digit expresses the nature of the response. This method allows clients to easily process responses without having to analyze the text of the response, which can also be localized into another language.

Answer types:

- A positive preliminary reply. The answer means that the server has accepted a request and has started processing it. No further action is expected from the client at the moment; the ball is in the server's court.
- A positive completion reply. The server has successfully completed the requested operation.
- A positive intermediate reply. The answer means that the server has accepted the request, but needs more information from the client in order to complete it. The client should know what information is needed according to the operation that was requested. For instance, during a user login operation, the client must respond to a 3xx response by sending a command containing a password. In other words – everything is fine, but the server is waiting for client activity.
- A transient negative completion reply. The operation failed; however, the error is not fatal. The usual reason is a temporary server overload. The client can repeat the request after some time without any change and there is a chance that the request will succeed.
- A permanent negative completion reply. This error is fatal and even resubmitting the same request will not succeed.

A similar mechanism was later adopted by a number of other protocols. In some of these (e.g. SMTP) the three-digit prefix has a similar meaning as in FTP, in others (e.g. HTTP) there are slight differences (surely at some point you have received a 404 response from an HTTP server – page not found).

Slide 62



Active/passive data connection

- Every data transfer uses new (data) connection
- Active data connection establishment:

```
1.1.1.1:1234 —— PORT 1,1,1,1,8,0 ——→ 2.2.2.2:21
             ←— 200 PORT command OK. ——
             —— LIST ——→
1.1.1.1:2048 ←·························· 2.2.2.2:20
```

- Passive data connection establishment:

```
1.1.1.1:1234 —— PASV ——→ 2.2.2.2:21
             ←— 227 OK (2,2,2,2,8,1) ——
             —— LIST ——→
1.1.1.1:1235 ··························→ 2.2.2.2:2049
```

- After data transfer closing, the connection is terminated

Introduction to Networking (2022) SISAL 62

Another aspect of FTP which overlaps with current protocols is the use of additional data channels.

In addition to the control connection, used only for sending client requests and server responses, special *data connections* are opened, through which all data content is transmitted. For each new transfer (a file download, file upload or directory listing), a new TCP connection must be opened, which will be closed again after the transfer is complete. But there are problems with opening additional data channels – both parties have to agree on **who** will open the data channel and to which **address and port** the connection should be established.

FTP distinguishes between two ways of opening a data channel, depending on the server’s role:

- An **active** data connection is established by the server. The server theoretically does not need any additional information – according to the original concept, it connects to the same address and port that the client uses. On the server’s side, the connection will use the server’s IP address, but it cannot use port 21 (all parameters of the new connection would be the same as the control connection and this is not possible), so another port 20 called *ftp-data* was reserved, which the server can use as the source port. An alternative, of course, is to use a generic port, like a regular client does when establishing a connection. However, the solution using the client's address and port as the destination address for the data connection is not entirely practical, so today most implementations initiate an active data connection with the PORT command (or EPRT for IPv6), which lets the server know another address and/or port. The syntax of the command may seem a little strange since the notation is a bit harder to read, but it is quite efficient – it's just six bytes of the socket address (4 bytes of the IP address and 2 bytes of the port number).

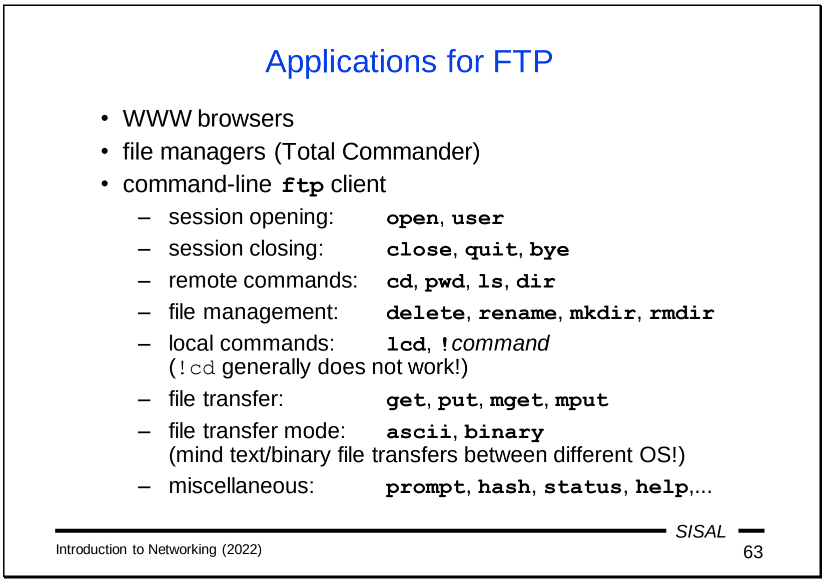
- In addition to the active method of establishing a data connection, there is also a **passive** method. In this case, the connection will be established by the client and will therefore need an address and port from the server. The client requests them with the PASV command (or EPSV for IPv6) and the server responds with a response that contains the required socket address in parentheses.

The proposed solution for establishing data channels corresponds to the state of network security in the seventies of the last century. All addresses were public and every address was accessible from anywhere. But at the moment when we start implementing security restrictions and using private addresses, the situation becomes more complicated. In the case of an active connection, the server will probably not be allowed to open a connection to the client, even if the client has a public address. However, the client is likely to have a private address today, to which the server cannot connect at all. The solution is to involve higher logic in the network address translation, where the NAT router must modify even the **content** of the messages between the client and the server and change the addresses sent in them accordingly. With a passive connection, the situation is slightly better: the server sends its public address to the client, so the only restriction may be if the router on the perimeter of our network allows opening the connection only to a **defined list** of ports.

The issue of opening additional data channels is not specific to FTP only. For instance, the SIP protocol, which is most often used today for voice transmission over a TCP/IP network, faces exactly the same problems.

Just as a reminder, the direction in which the data connection is being opened is in no way related to the direction in which the data will then flow.

Slide 63



Applications for FTP

- WWW browsers
- file managers (Total Commander)
- command-line **ftp** client
  - session opening: **open**, **user**
  - session closing: **close**, **quit**, **bye**
  - remote commands: **cd**, **pwd**, **ls**, **dir**
  - file management: **delete**, **rename**, **mkdir**, **rmdir**
  - local commands: **lcd**, **!***command*
    (!cd generally does not work!)
  - file transfer: **get**, **put**, **mget**, **mput**
  - file transfer mode: **ascii**, **binary**
    (mind text/binary file transfers between different OS!)
  - miscellaneous: **prompt**, **hash**, **status**, **help**,...

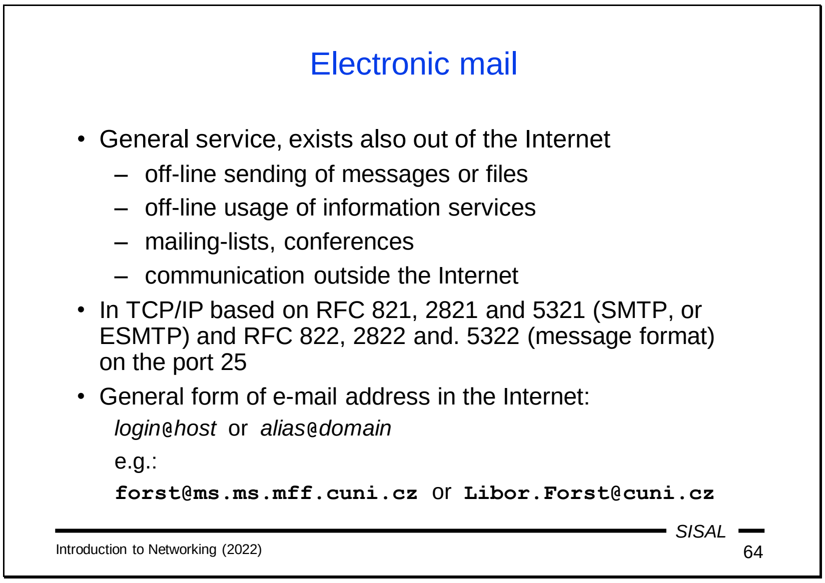
SISAL

Introduction to Networking (2022) 63

You can currently use the following ways to access the FTP protocol:

- Web browsers. To download a file or list a directory, just enter the appropriate URL; for uploading a file, it is often necessary to install an extension.
- File managers. Many file management applications can work with FTP. For instance, Total Commander can display the contents of a directory on an FTP server (instead of the contents of a local disk) in a panel and perform common operations with files. For the study of FTP, it is interesting that in an additional window it is possible to monitor the FTP commands by which individual operations are performed.
- The last resort is the oldest application, the interactive **ftp** program, which accepts line based commands. Most of them are logical, you just need to be careful when transferring files between operating systems that use different line endings of text files (e.g. Windows vs. UNIX) – unless we choose the correct transfer mode (**ascii** for text files and **binary** for binary ones), the files on the target computer will be unreadable or damaged.

Slide 64



Electronic mail

- General service, exists also out of the Internet
  - off-line sending of messages or files
  - off-line usage of information services
  - mailing-lists, conferences
  - communication outside the Internet
- In TCP/IP based on RFC 821, 2821 and 5321 (SMTP, or ESMTP) and RFC 822, 2822 and. 5322 (message format) on the port 25
- General form of e-mail address in the Internet:

  login@host or *alias@domain*

  e.g.:

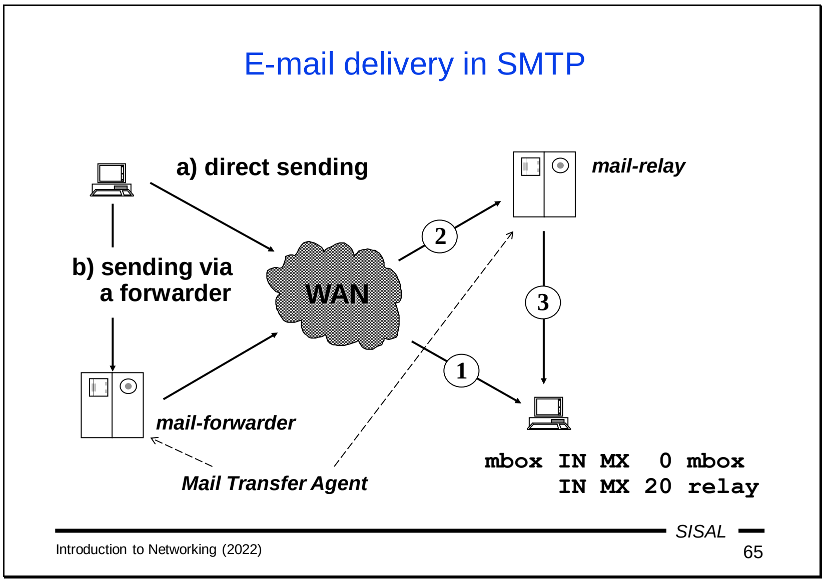
  `forst@ms.ms.mff.cuni.cz` or **`Libor.Forst@cuni.cz`**

Electronic mail is another very old service, dating back to before the “modern Internet”. Already in computer prehistory, it served as one of the few communication bridges between computer networks of various kinds, and the variety of postal address formats also reflected this. At present, a form composed of two parts separated by an “at” symbol (“@”) become standard on the Internet. In its basic form, the at symbol is preceded by a mailbox name and followed by the name of the server where the mailbox is located. However, this form is both too dependent on the current state (changing the server name would change the email addresses of all of its users) and poses a risk, because a potential attacker can read both the server name and the account name from the address and can try to attack the account. Therefore, for both reasons, an alias is more often used instead of an account name, and a domain or service name is used instead of a specific server name.

The original concept of email was to serve for the transmission of short messages (up to 64 kB), either for personal correspondence or for correspondence within a certain group of people (a mailing list), but also as an off-line method of using various services (e.g. searching for documents in FTP archives). Over time, however, email has begun to be used more frequently also for file transfer, which brings some complications.

The Internet uses the SMTP protocol for mail transmission. It is a text based protocol on TCP port 25 with the principle of sending messages and replies, similar to FTP.

Slide 65



E-mail delivery in SMTP

a) direct sending

b) sending via a forwarder

WAN

mail-relay

mail-forwarder

Mail Transfer Agent

1

2

3

```
mbox IN MX  0 mbox
     IN MX 20 relay
```

Introduction to Networking (2022)

SISAL

65

Let's now look at what happens when a user issues a command to send an email message.

The mail program checks the part of the address after the at symbol and finds out which server receives mail for the given domain. Theoretically, it is possible that there are no obstacles in the path between the client computer and the destination server, so the program can establish an SMTP connection to the destination server and try to forward the message directly. However, this is not typical; usually the message's path includes several nodes which pass the message to each other in turn.

On the sender side, the reason is usually that the mail program has a simplified configuration and instead of complex rules, it only indicates that mail is to be forwarded via SMTP to a special server on the local network (a so-called *mail-forwarder*) intended for further mail processing. This step is called *mail submission*. There may be even more such forwarders, e.g. due to several levels of NAT.

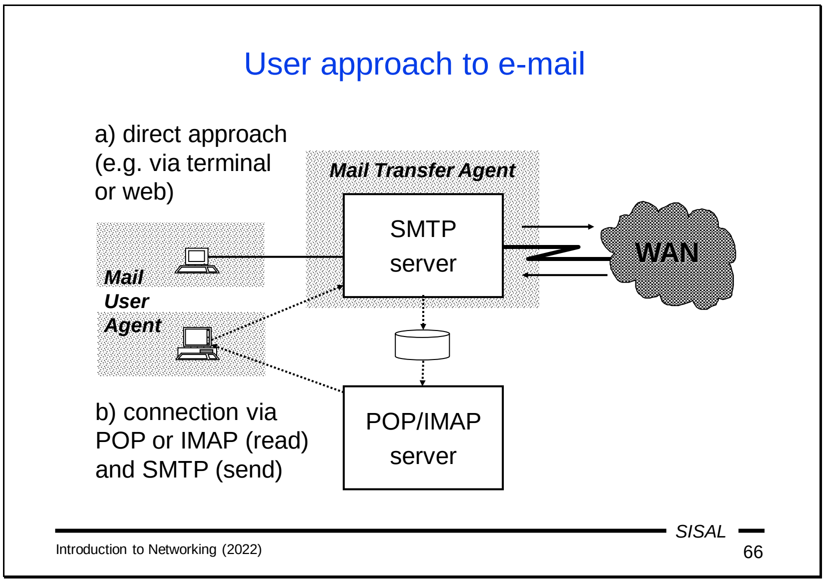
Each node that receives and delivers mail is called a Mail Transfer Agent (MTA). Individual MTAs forward the message using the SMTP protocol, where the sending MTA temporarily acts as a client and the receiving MTA as a server. If the server is not the final destination, it queues the message and periodically attempts to deliver it to the next MTA. Note that the number of MTAs along the way has nothing to do with the number of networks or routers.

If the destination mailbox is on a server that is freely accessible from the Internet, the last MTA will try to deliver the mail to that server. If delivery is not possible, the mail remains in the queue on the last MTA (generally "somewhere on the Internet"). If the mail server administrator wants to prevent this, he can set up a DNS MX record for his server. MX records contain the name of the *mail exchanger* that is temporarily or permanently authorized to receive mail for a given machine or domain plus its priority (the lower the number, the higher the priority). In such a case, the delivering MTA sequentially tries to contact individual exchangers, according to priorities, until it succeeds.

The recipient can follow the entire path of the message over the Internet using the Received headers in the delivered message.

Given how messages are delivered, it is probably clear why this protocol is not suitable for transferring large files. Each MTA node on the path must store the entire message in its queue just to pick up it from the queue a while later and send it on.

Slide 66



User approach to e-mail

a) direct approach (e.g. via terminal or web)

Mail Transfer Agent

SMTP server

WAN

Mail User Agent

POP/IMAP server

b) connection via POP or IMAP (read) and SMTP (send)

Introduction to Networking (2022)

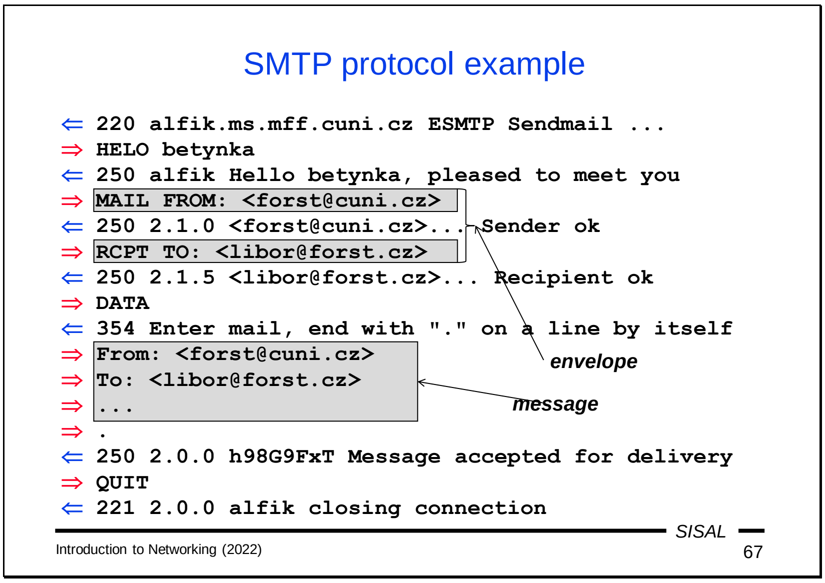
SISAL

66

From the user's point of view, the mail system is accessible through a "mailer program“, a so-called Mail User Agent (MUA). In principle, there are two ways in which an MUA is connected to the mail system:

- The first option is a direct connection. The user connects from his computer to the MTA, where he has his mailbox. Various connection methods are possible, such as a remote login to the system, or visiting a web application running on the mail server, and the user authentication options also correspond to the type of connection method. In this case, the application itself has direct access to the messages in the user's mailbox and at the same time to the local services of the MTA, so it directly places messages to be sent in the MTA queue.
- An alternative is to connect to the mailbox using one of the special mail protocols, POP or IMAP. A client of such a protocol connects to a server running on the MTA, authenticates itself using this protocol and, based on user requests, sends commands to the server for handling the mailbox. However, these commands only work with received messages. For sending messages, the client must submit mail normally using SMTP, not necessarily to the same server. Therefore, we will also find separate sections for POP/IMAP and SMTP in the client application configuration. For example, in a situation where a user travels and connects to his own server "from the outside", his server may not be willing to receive a message from him due to lack of authentication in SMTP, so the user may not be trustworthy enough for the server. This can be solved, for example, by temporarily using a local MTA of the ISP to send mails.

Slide 67



SMTP protocol example

```
⇐ 220 alfik.ms.mff.cuni.cz ESMTP Sendmail ...
⇒ HELO betynka
⇐ 250 alfik Hello betynka, pleased to meet you
⇒ MAIL FROM: <forst@cuni.cz>
⇐ 250 2.1.0 <forst@cuni.cz>... Sender ok
⇒ RCPT TO: <libor@forst.cz>
⇐ 250 2.1.5 <libor@forst.cz>... Recipient ok
⇒ DATA
⇐ 354 Enter mail, end with "." on a line by itself
⇒ From: <forst@cuni.cz>
⇒ To: <libor@forst.cz>
⇒ ...
⇒ .
⇐ 250 2.0.0 h98G9FxT Message accepted for delivery
⇒ QUIT
⇐ 221 2.0.0 alfik closing connection
```

Introduction to Networking (2022) SISAL 67

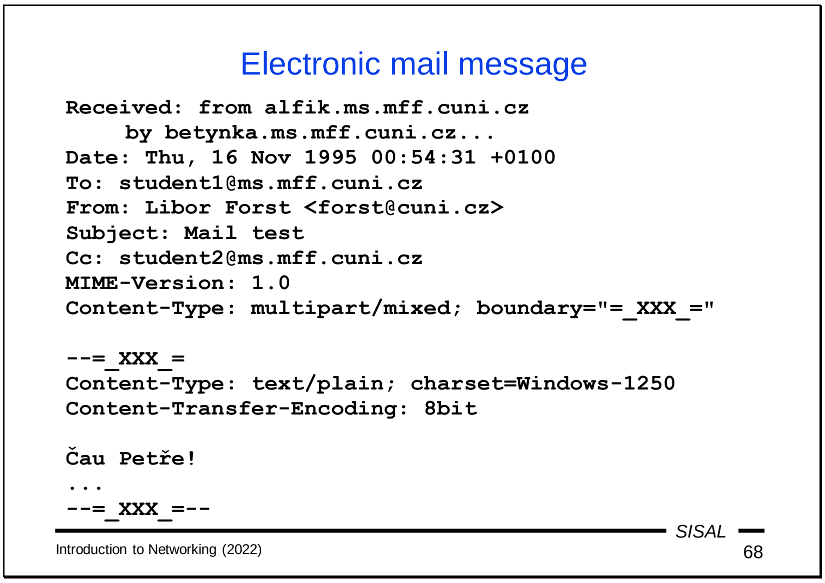
The SMTP protocol is similar to FTP: the client sends individual commands as text lines and the server responds similarly, even including a three-digit code with the same meaning of the first digit.

Sending a new message begins with the client's MAIL FROM command, which includes the sender's address as a parameter (in angle brackets). This command is followed by one or more RCPT TO commands, each with an address of one recipient. The server confirms each recipient separately – if it decides not to accept one, it responds with message code 450 (temporary error) or 550 (permanent error) instead of message code 250. However, if it answers with 250, he **takes over the responsibility** for the delivery of the message or the sending of a message about a delivery failure, a so-called Delivery Status Notification (DSN). The command with the last recipient is followed by a DATA command, to which the server responds with message code 354, after which the client starts sending the text of the message, i.e. roughly what the recipient will then see in his mailbox. The protocol is textual, because the messages were originally so intended, and ends with a line that contains only a dot itself. Therefore, SMTP clients must pay attention to the mail lines that start with a dot (or binary bytes which coincidentally form the sequence CR LF dot!). They send such lines to the server with one extra dot prepended, which the server deletes when saving the message. The server responds to the terminating line with a standard response (250, 450, or 550), and the client can finish with a QUIT command or continue with another message.

Note 1: What relationship is there between the addresses that the client sends to the server as the sender's and recipients' addresses and those that the recipient then sees in the message? **There is none!** The addresses used in the protocol are called the *envelope*, and their relationship to the body of the message is the same as that of the address written on a real mail envelope and the addresses written on the message inserted in the envelope. It is simply just text that the sender can fake; however, unfortunately, your mail program is then driven by it…

Note 2: An MTA that accepted a message (i.e. replied with code 250 to the specific addressee and to the body of the message as well), but who failed to deliver the message or pass it on to another MTA, has a duty to generate a DSN (Delivery Status Notification) message. Because the sender in this case is the mail system itself, no address is filled in the MAIL FROM statement. Such messages tend to be easier to pass on some nodes (for example, they are less rigorously checked for viruses or spam), which is why some spam machines abuse this method of sending. Unfortunately, this leads some administrators to simply disable DSN delivery. However, this is a gross violation of the rights of their users, because the entire e-mail system is built on the principle of *best effort*. The key element of it is that the sender will be notified of any failure of delivery.

Slide 68



Electronic mail message

```
Received: from alfik.ms.mff.cuni.cz
     by betynka.ms.mff.cuni.cz...
Date: Thu, 16 Nov 1995 00:54:31 +0100
To: student1@ms.mff.cuni.cz
From: Libor Forst <forst@cuni.cz>
Subject: Mail test
Cc: student2@ms.mff.cuni.cz
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="=_XXX_="

--=_XXX_=
Content-Type: text/plain; charset=Windows-1250
Content-Transfer-Encoding: 8bit

Čau Petře!
...
--=_XXX_=--
```

Introduction to Networking (2022) SISAL 68

An electronic mail message consists of two parts separated by an empty line.

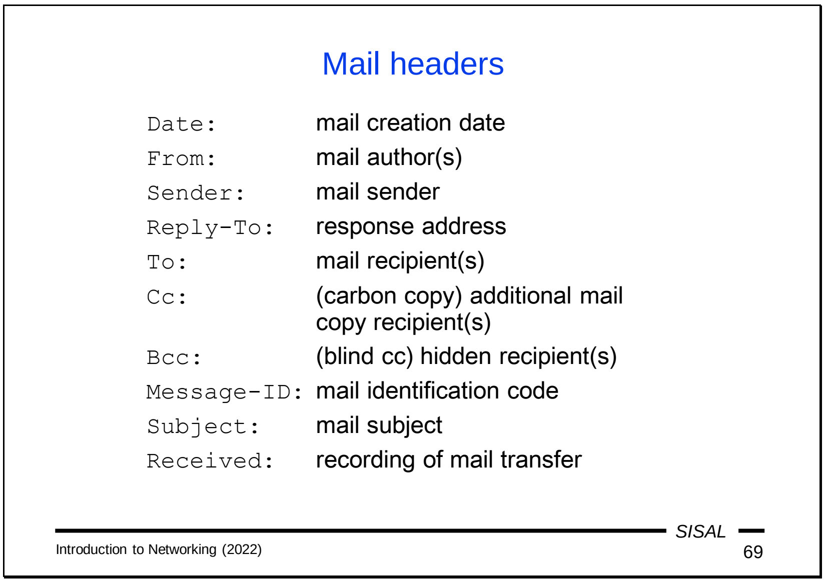
The first part contains *headers*, lines with control information, with a relatively fixed format and characters exclusively from the ASCII table (i.e. only 7-bit characters 0x00 to 0x7f, no characters of national alphabets). Headers are intended for both end-user information and the operation of e-mail applications, which usually display only a selected portion of the headers to the user, but tend to have an operation or setting that allows the user to see the entire header list.

The second part is the text of the message. In the original concept, it was possible to use only ASCII characters and write only plain text.

Over time, however, the possibility of using non-Latin characters in the text of messages became popular. There is an extension of the protocol called ESMTP, in which the client and server can agree that both are able to accept 8-bit characters – the client can then use the so-called **8-bit** content transfer encoding.

The second major innovation was the ability to define the **structure and semantics** of the message body using the MIME extension. Thanks to this, it was possible to start working, for example, with attachments containing files.

Slide 69



Mail headers

`Date:`	mail creation date
`From:`	mail author(s)
`Sender:`	mail sender
`Reply-To:`	response address
`To:`	mail recipient(s)
`Cc:`	(carbon copy) additional mail copy recipient(s)
`Bcc:`	(blind cc) hidden recipient(s)
`Message-ID:`	mail identification code
`Subject:`	mail subject
`Received:`	recording of mail transfer

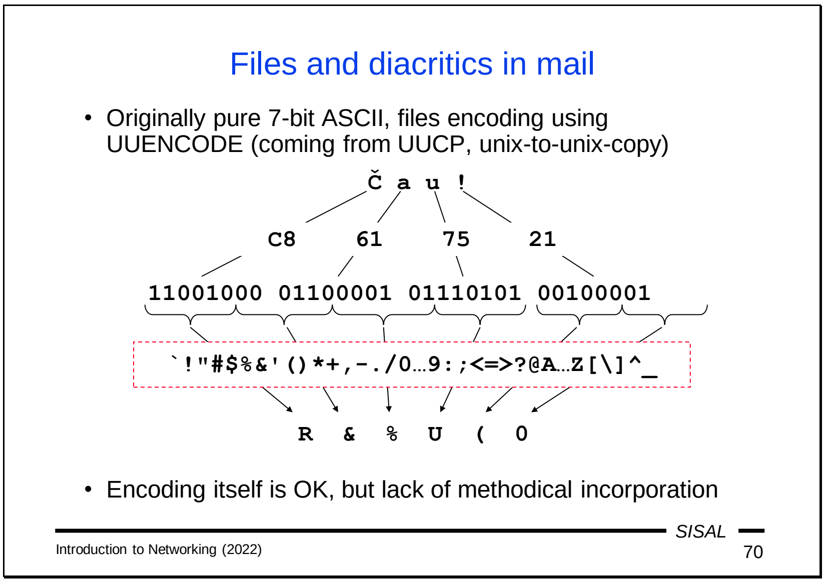
The most important mail headers include:

- Date – contains the date of origin of the message in a fixed (American) format.
- From – contains the address of the author (or authors) of the message; either a simple e-mail address or an address accompanied by a comment (usually a full name), most often in the format "comment <address>" or "(comment) address". The header was really intended as a list of authors, that is, if the message is written by more people, everyone should be listed here. But mail applications usually don't have any sophisticated support for this, so users ignore it and usually write only one author in the From header. On the contrary, mentioning more "authors" is often a symptom of spam - spam machines are based on the dubious assumption that if they use a "known address" in From, various mail checking automata will be more tolerant, and therefore they commonly have a long list of addresses in the From header.
- Sender – is, on the other hand, a header where there can be only one address. It indicates who in the From list has actually sent the message, or possibly a sender who is not actually in the From list (e.g. a secretary sending a message for her boss).
- Reply-To – contains the address that the recipient should use when sending a reply (e.g. the address of the mailing list from which the message came).
- To – contains a list of message recipients.
- Cc – contains a list of recipients to whom a copy of the message is sent. Logically, these should be people who are not expected to take part in the dialogue and are just "in copy". Note, however, that the average user does not look at whether he or she has been listed as To or Cc when receiving a message. Therefore, it is not a good idea to write a message to your friends starting with "Hey, guys" and send it Cc-ed to Mr. Dean. A practical problem is that many mail applications mix the original To and Cc recipients when creating the response. The name of the header

is derived from the name of carbon copy paper used to make copies when typing on a physical typewriter.

- Bcc – ("blind carbon copy") is a header that does not appear in the actual message, it is displayed only by the mail program and the recipient is added to the SMTP envelope, but not to the text. Therefore, the other recipients do not know that the person was also among the addressees.
- Message-ID – is a technical header produced by the sending mail program. The recipient's program should use it when preparing a response. It allows the logical grouping of answers into discussion "threads".
- Subject – contains a brief summary of the content of the message (such as the "Subject“ used in business messages).
- Received – is a technical header, which is usually added to the message by each node that forwards it. The header contains the name of the MTA, the time of delivery, the local identifier and possibly other data that may facilitate the tracing of the message.

Slide 70



Files and diacritics in mail

- Originally pure 7-bit ASCII, files encoding using UUENCODE (coming from UUCP, unix-to-unix-copy)

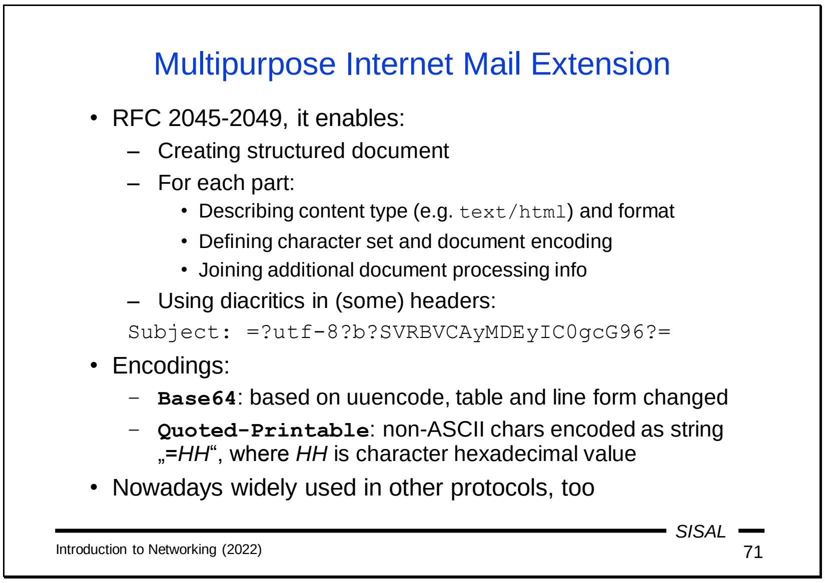
- Encoding itself is OK, but lack of methodical incorporation

SISAL

Introduction to Networking (2022)

As already mentioned, in the original email proposal it was possible to use only ASCII characters in the entire message. However, users sometimes needed to transfer files that were binary. Users of the UNIX-to-UNIX copy (UUCP) network already had a similar problem at the time. They also needed to send files over a 7-bit channel and they invented the UUENCODE encoding that allowed them to do so. In this encoding, three bytes of the original file are taken, these 24 bits are divided into four groups of six bits, and these six-bit codes are converted to four printable characters using a fixed table. The encoded file is therefore 33 % larger than the original. UUENCODE was created at a time when it was not common to distinguish between lowercase and uppercase messages, so the table has only 26 (uppercase) messages, 10 digits and the remaining 28 places are occupied by almost all other characters, including those with special meanings in various environments (a semicolon, an apostrophe, a quotation mark, an asterisk, square brackets, at symbol, etc.). However, the encoding already existed, and it was possible to insert an encoded file into the text of an email message merely by surrounding it with begin and end lines. The main problem with this solution was that the structure of a message, i.e. whether it contained embedded files and the attributes of those files, could be determined only by analyzing the entire text of the message.

Slide 71



Multipurpose Internet Mail Extension

- RFC 2045-2049, it enables:
  - Creating structured document
  - For each part:
    - Describing content type (e.g. `text/html`) and format
    - Defining character set and document encoding
    - Joining additional document processing info
  - Using diacritics in (some) headers:

```
Subject: =?utf-8?b?SVRBVCAyMDEyIC0gcG96?=
```

- Encodings:
  - **`Base64`**: based on uuencode, table and line form changed
  - **`Quoted-Printable`**: non-ASCII chars encoded as string „=*HH*“, where *HH* is character hexadecimal value
- Nowadays widely used in other protocols, too

The problem of structuring email messages was solved by the Multipurpose Internet Mail Extension (MIME), which was later, despite its name, used in many other protocols (e.g. HTTP).

Every document in MIME format contains a header and a body (similar to the mail itself), where the header defines attributes such as:

- a document type; MIME types consist of a main type plus a subtype (e.g. text/html, image/jpeg, audio/mp3, application/pdf),
- the character set used,
- an encoding method,
- the original file name,
- the intended method of processing (to display as an attachment or to insert into the text)…

However, one of the key document types is **multipart**, which means that the body of the document is further structured. The sending application generates a random string, which is used as a separator for each part of the structured document, and each of these parts is again a standard MIME document. If you attach two files to a message in a typical current mail program, the message will be sent as a MIME multipart document, where the first part will be the message text and the other parts will be the attached files.

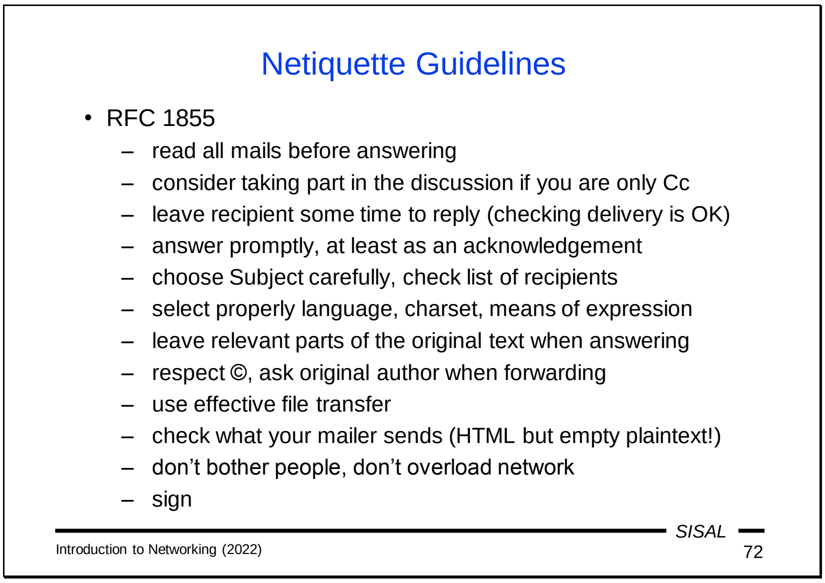
MIME defines two basic encoding methods:

- The method called **Base64** is essentially nothing more than the old known UUENCODE. MIME only replaced the original coding table with a more modern one (today's computers do not have a problem with lowercase and uppercase messages, which gives us 52 characters instead of 26 and if we add digits, we need to add only two non-alphanumeric characters, plus and slash) and slightly changed the line format.

- The **Quoted-Printable** method is mainly used for text files, where most of the text consists of ASCII characters. Each non-ASCII character is converted to the sequence "=HH", where HH is the hexadecimal value of the character (e.g. the equal sign itself must be encoded as "=3D"). The advantage over Base64 is that the text remains at least a little bit readable. From the point of view of coding efficiency, the ratio of ASCII and non-ASCII characters is important – if we encode text containing only non-ASCII characters, we will get a code size of 300 % of the original (compared to fixed 133 % for Base64), and for a pure ASCII text we get only slightly above 100 %.

In addition to the advantages for transferring documents, MIME also brought the ability to insert non-ASCII characters into (some!) headers (e.g. the Subject). The encoding starts with the characters "=?" followed by an identifier of the character set used, a question mark, an identifier of the encoding ("b" or "q"), a question mark, the encoded text, and the final sequence "?=".

Slide 72



Netiquette Guidelines

- RFC 1855
  - read all mails before answering
  - consider taking part in the discussion if you are only Cc
  - leave recipient some time to reply (checking delivery is OK)
  - answer promptly, at least as an acknowledgement
  - choose Subject carefully, check list of recipients
  - select properly language, charset, means of expression
  - leave relevant parts of the original text when answering
  - respect ©, ask original author when forwarding
  - use effective file transfer
  - check what your mailer sends (HTML but empty plaintext!)
  - don’t bother people, don’t overload network
  - sign

SISAL

Introduction to Networking (2022) 72

The style in which e-mail should be used is subject to certain rules. Some of them are similar to the general rules of non-electronic communication, some are completely new. Their nature is a recommendation, but most of them should be really followed under normal circumstances. The specificity of electronic communication has even led to the most important rules being written in the form of an RFC.

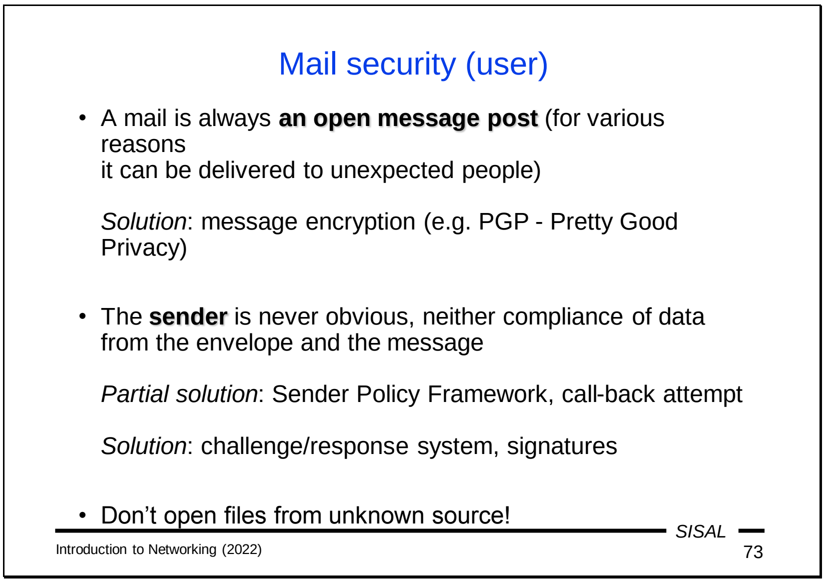
So here is how a user should behave:

- When you open a mailbox with incoming messages, you should first look at all the messages received and only then start replying to them. It often happens that the discussion in a thread has already progressed in such a way that your answer to the first message in the sequence would be useless or confusing.
- The RFC says that if you are only a Cc recipient, you should carefully consider whether you want to actively intervene in the discussion. The problem with current mail programs, however, is that after several replies, you usually don't know if you were originally on the Cc or To list. But you should usually know this from the nature of the dialogue.
- If you send someone a question or request, you should give them time to answer. Mail is an off-line service and you do not know the recipient’s situation, how soon he or she will get to reading the mail. Of course, you can, for example, send a decently worded message the next day, asking at least for a confirmation that the message has arrived, because you are never absolutely sure about this. Of course, it is also possible in an emergency to contact the recipient in another way and notify him of your message.
- A complementary rule, however, is that you should respond quickly. It means that after reading the message, decide whether you are able to reply within a reasonable time (depending on the nature of the message, it is usually several

hours, no later than the next day), and in the negative case at least acknowledge receipt of the message with an estimation of when you will be able to reply.

- Work with the Subject header. Try to express the essence of the message in a few words so that the recipient knows how important it is and what it is about. A message with the Subject "Query" is the same as a message without a Subject. If the nature of the message changes during the reply, change the Subject.
- Check the mailing list carefully to confirm that you are sending the message to everyone you wanted, and that you are not sending it to someone to whom it should not be sent.
- Carefully select the language and style of the message according to the nature of the content and the list of recipients (think of Cc recipients as well).
- When sending a reply, consider whether it is necessary to enclose the original message (if your reply is "Yes", it is probably necessary) and whether it is necessary to enclose it in its entirety. Especially if the original message is long and you only respond to a few sentences from it, consider copying only the relevant sentences into your answer.
- Respect the rights of other participants in the discussion. You are not formally entitled to forward a text or image of another author to a third party without the author's consent. Of course, you won't worry about this among a group of friends, but when the communication is more official, you have to consider it.
- If you are going to send a larger file to multiple recipients, consider whether really almost all of them need it. Otherwise, save the file in a convenient location (your website, Dropbox, etc.) and send only a link.
- Even if your message does not contain a large file, but has many recipients, carefully consider whether you really want to send it to everyone. A separate chapter is, of course, the case of chain messages.
- Review the content of your message and adjust the format accordingly. A frightening case is when someone creates a document (e.g. in Word) containing two sentences and attaches it as a file to a message, instead of writing the two sentences directly in the message itself. Current e-mail applications approach the problem similarly and often send a message as an HTML document even when it is not absolutely necessary. This can be a problem for some recipients, especially if the mailer sends a multipart/alternative MIME document, which means that the message contains two interchangeable variants, one of which is of type text/html and the other of which is text/plain, which is empty!
- Sign the message.

Slide 73



Mail security (user)

- A mail is always **an open message post** (for various reasons
it can be delivered to unexpected people)

  Solution: message encryption (e.g. PGP - Pretty Good Privacy)

- The **sender** is never obvious, neither compliance of data from the envelope and the message

  Partial solution: Sender Policy Framework, call-back attempt

  Solution: challenge/response system, signatures

- Don’t open files from unknown source!

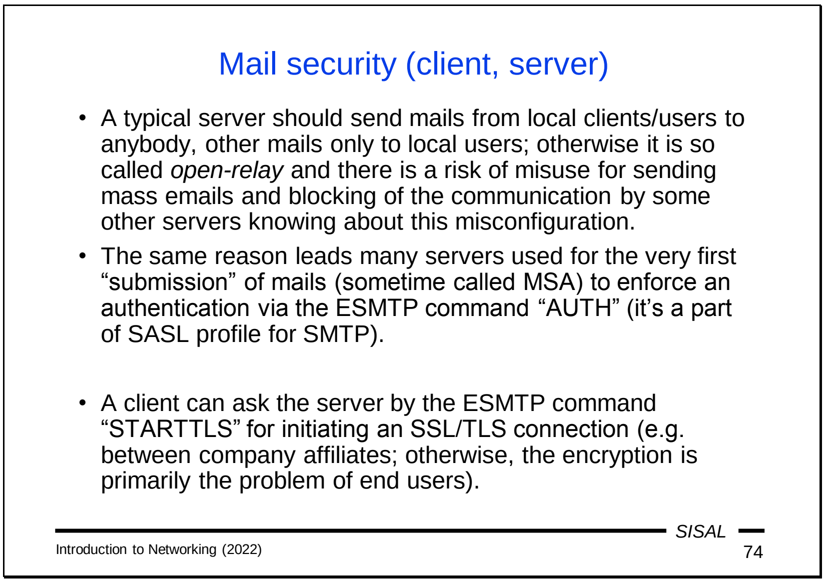
Introduction to Networking (2022) SISAL 73

An email user should keep in mind that the content of a message is not protected during transport and due to technical errors or an attack, it may get into the hands of people for whom it was not intended. Not to mention the possibility that the sender may make an error when typing the recipient's address. The sender should adapt the content of the message accordingly. And if the content of the message is confidential, the user should use some encryption system.

It is also clear that thanks to the open mail protocol, anyone can write anything into the envelope or into the message. The recipient is never sure who the real sender is, and he/she should never blindly believe what is written in the message. As we have already said, if it is not serious, the recipient can probably estimate the authenticity according to the style of the text, but it is necessary to be careful. If a friend sends me a file that we talked about at lunch the day before, I'll probably believe it's from him. However, if he sends a file without notifying me about it and if I cannot tell from the text that it is genuine, I will **definitely not open** the file until, for example, I call him.

Mail servers intended for receiving mails for end users attempt to prevent mails from an insecure sender (for example, by trying to send a DSN message to that sender), but such protection is not entirely reliable. Some automated systems use the well-known challenge/response method, but only an electronic signature provides fully reliable protection.

Slide 74



Mail security (client, server)

- A typical server should send mails from local clients/users to anybody, other mails only to local users; otherwise it is so called *open-relay* and there is a risk of misuse for sending mass emails and blocking of the communication by some other servers knowing about this misconfiguration.
- The same reason leads many servers used for the very first “submission” of mails (sometime called MSA) to enforce an authentication via the ESMTP command “AUTH” (it’s a part of SASL profile for SMTP).
- A client can ask the server by the ESMTP command “STARTTLS” for initiating an SSL/TLS connection (e.g. between company affiliates; otherwise, the encryption is primarily the problem of end users).

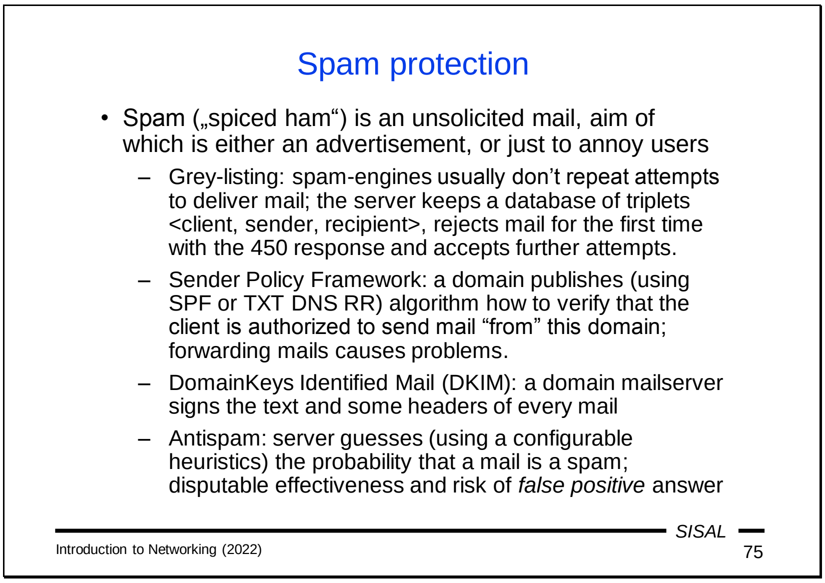
Introduction to Networking (2022) SISAL 74

A properly configured mail server should distinguish between mails sent by „its" users, which it should deliver without restriction, and mails coming "from the outside", which should only be accepted if they are addressed to some of „its" users. If the server allows anyone to connect and send a mail anywhere, it is called an *open-relay server* and runs the risk of being misused to send bulk mail. Spammers try to cover their tracks by forwarding mail via several open-relay servers, so finding the real originator is complicated. Conversely, there are organizations that scan servers around the world, look for those that are open-relay, and compile a list of them that mail server administrators can use to block the receipt of mails from any of the open-relay servers.

The problem occurs when one of the server’s own users needs to connect to his mail server "from the outside", because he is just travelling outside the local network. From the server's point of view, it is a foreign client and sending the mail will be rejected. And the user has no way to prove his identity to the server, because the SMTP protocol does not use authentication by itself. However, there is an extension that the server can use (usually on a different port, 587) to force authentication.

Since the transmission of a message between the sender and the recipient is not a matter of a single connection, but the message is stored several times along the way, the question of securing the content of the message is entirely a matter for the end users. Therefore, the connection between MTAs is usually not encrypted in any way. However, there are exceptions – e.g. if we have two corporate mail servers connected by a public network, it makes sense to encrypt the connection between them so that all corporate mails do not have to be encrypted by their senders. The ESMTP extension includes a STARTTLS command that asks the server to start encryption using the Transport Layer Security interlayer.

Slide 75



Spam protection

- Spam („spiced ham“) is an unsolicited mail, aim of which is either an advertisement, or just to annoy users
  - Grey-listing: spam-engines usually don’t repeat attempts to deliver mail; the server keeps a database of triplets <client, sender, recipient>, rejects mail for the first time with the 450 response and accepts further attempts.
  - Sender Policy Framework: a domain publishes (using SPF or TXT DNS RR) algorithm how to verify that the client is authorized to send mail “from” this domain; forwarding mails causes problems.
  - DomainKeys Identified Mail (DKIM): a domain mailserver signs the text and some headers of every mail
  - Antispam: server guesses (using a configurable heuristics) the probability that a mail is a spam; disputable effectiveness and risk of *false positive* answer

SISAL

Introduction to Networking (2022) 75

Spam has become a very unpleasant part of electronic mail communications. Protection against it is difficult because any restriction will partly affect real messages sent by real users.

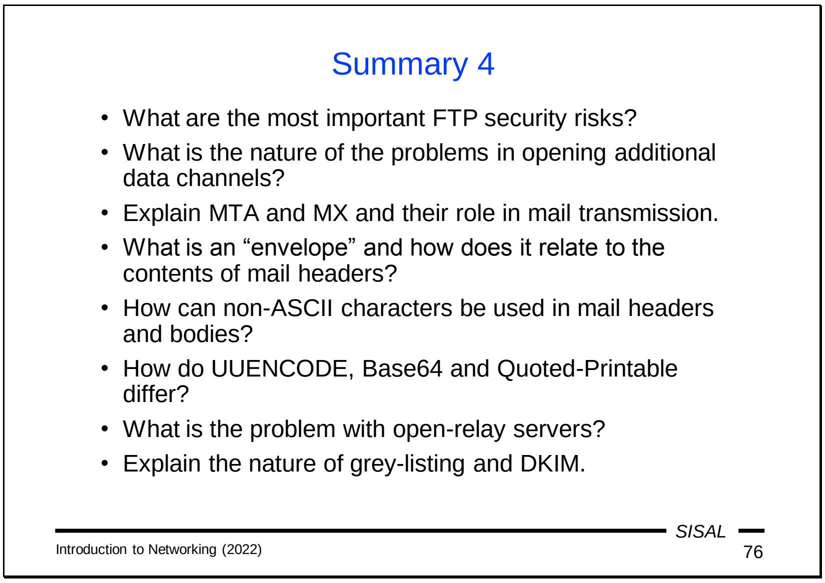
The **grey-listing** method uses protocol properties and the experience that spam machines do not check delivery success (there are certainly a number of invalid addresses among a huge number of recipients' addresses, but they form an insignificant part of the overall effect, so it makes no sense for spammers to detect and repeat unsuccessful delivery). The server will therefore respond to the first attempt to deliver a message to a recipient, from a sender, and coming from an IP address with the 450 response to the RCPT TO command. If it was a genuine message, the client will attempt a new delivery after a certain time (typically 15 minutes), the server will discover that there has been a repeated attempt, and the new request will be answered normally with a 250 response and the message will be delivered. At the same time, it will mark the time of the last delivery attempt on a given triplet, and for some time (e.g. 5 weeks) it will pass all subsequent messages without a delay while each subsequent attempt shifts this time stamp. As a result, if two partners exchange at least one message a month, their communication is carried out without delay, except for the 15-minute delay of the first message. If, on the other hand, there is no repeated attempt after the first one (e.g. within 1 hour), the triplet will be blacklisted for a period of time. It was the combination of black and white list ideas that gave the method its name.

Another way to solve the problem was the so-called Sender Policy Framework. The basic idea is simple: a domain defines a list of servers it uses to send mails, and no server in the world should accept mails whose sender is someone from that domain coming from a client not listed here. Relatively complex syntax rules were developed on how to define the correct sending MTAs, and these were formerly published using the DNS TXT record, later even with the newly created SPF record. But the catch to the principle is that once a message arrives somewhere where a user has set forwarding of incoming mails to a different location, the verification fails because the message with the original sender is suddenly being sent by a different machine! While correction mechanisms were established, the entire system proved to be inflexible, so that the DNS SPF type has been even withdrawn.

However, a more successful replacement was created, the so-called DomainKeys Identified Mail. The basic idea is the same; the difference is that a sending machine **digitally signs** all mails. More specifically, it signs the mail body and selected headers. The receiving server checks the signature and, as a result, either delivers the message to the destination mailbox or rejects it. Compared to SPF, this solution has the advantage that forwarding does not affect the signature in any way. All sending MTAs for a given domain can be equipped with a separate key, so that sending can be distributed to multiple machines and the list of them changed flexibly.

Various **spams blockers** are also a common means of protection. These are algorithms that attempt to estimate the probability that a message is spam. They use a number of attributes that a local administrator can give its own weights to. Attributes relate to both form (e.g. Subject in capital messages) and content (e.g. the frequency of some words). Based on the resulting score, the message can either be completely discarded, quarantined, or just tagged and delivered. Unfortunately, these algorithms are debatable in their effectiveness, because spam machines are increasingly adept at mimicking real messages, while strict rules increase the percentage of wrongly misclassified messages (so-called *false positive* outcomes).

Slide 76



Summary 4

- What are the most important FTP security risks?
- What is the nature of the problems in opening additional data channels?
- Explain MTA and MX and their role in mail transmission.
- What is an “envelope” and how does it relate to the contents of mail headers?
- How can non-ASCII characters be used in mail headers and bodies?
- How do UUENCODE, Base64 and Quoted-Printable differ?
- What is the problem with open-relay servers?
- Explain the nature of grey-listing and DKIM.