

6 Application Layer



arrangement

6 Application Layer

6.1 Classification of Applications

6.2 World Wide Web

6.3 E-Mail

6.4 Domain Name System

6.5 Exercises - Application Layer

6.6 Summary - Application Layer

As already discussed in the introduction to the OSI Reference Model, no distinction is made between layers 5, 6 and 7 in practice. Therefore, this chapter only refers to a single Application Layer. It accesses the services of the Transport Layer.

Because of the large number of network applications available, it is necessary to select only a few. So only the applications WWW (protocol HTTP), e-mail (protocols SMTP, POP, IMAP) and DNS are presented here. At the beginning of the chapter, a classification of applications is given.

6.1 Classification of Applications

Applications can be distinguished according to the requirements they have related to the network. In this respect, one can distinguish, in particular, how well the applications can handle transmission errors and delays in transmission. The following table shows an overview.



Classification of applications

Begin printversion

	Interactive (delay << 1 s)	Responsive (delay approx. 2 s)	Timely (delay approx. 10 s)	Not relevant (delay >> 10 s)
Error tolerant	audio and video conversations (VoIP)	voice and video messages	audio and video streaming	fax
Error intolerant	interactive terminal (telnet/SSH), interactive games	transactions (e.g. e-commerce, web surfing, e-mail retrieval)	text messages, downloads (e.g. FTP)	e-mail transfer

End printversion

6.2 World Wide Web



arrangement

6.2 World Wide Web

6.2.1 URI: URL and URN

6.2.2 Hypertext Markup Language

6.2.3 Tasks of Client and Server

6.2.4 HTTP

6.2.5 HTTP/2

The World Wide Web was developed in 1990 by Tim Berners-Lee at CERN (European Center for Nuclear Research) in Geneva to exchange information between scientists (see [CERN page](#)). He programmed the first browser and web server as part of this. Between 1993-1995, the browser [Mosaic](#) led to the widespread use of WWW (a further development of this browser was Netscape Navigator, which is the forerunner of today's Firefox). Together with the abolition of commercial restrictions, this application resulted in rapidly increasing user numbers for the Internet, which also made it interesting for people without scientific background.

The development of technical specifications related to WWW has been carried out since 1994 by the W3C ([World Wide Web Consortium](#)), whose chairman is Tim Berners-Lee. The W3C does not call them standards but instead talks about recommendations. Some selected examples of languages defined by the W3C are HTML, CSS, XML, PNG, SVG, XQuery.

The WWW itself is defined in the ([W3 recommendations](#)) as follows: "The World Wide Web" (WWW, or simply Web) is an information space in which the objects of interest are referred to as Uniform Resource Identifiers (URIs)."

The Hypertext Transfer Protocol ([HTTP](#)), which is used to deliver the WWW pages, is defined by IETF.



task

Task: WWW versus Internet

Is there a difference between the terms WWW and Internet?

Solution

Yes, the Internet consists of networks in which computers use the Internet Protocol. This definition thus belongs to the third OSI layer. The World Wide Web, on the other hand, is an application (OSI Layer 7), which is realized with the help of the Internet. If, for example, you receive e-mails with a program such as Thunderbird, you use the Internet but not the WWW.

6.2.1 URI: URL and URN

In the definition of the WWW, the term URI was used. So this term is presented in more detail here.

- A **uniform resource identifier (URI)** consists of a string of characters and identifies an abstract or physical resource. URI is a general term that can be further classified as a “locator” or name.
- A **uniform resource locator (URL)** is a subset of a URI and is used if a resource is accessed through its access mechanism, i.e. its location.
- A **uniform resource name (URN)** specifies a name that is globally unique and persistent; that is, it persists even if the resource is no longer available or is no longer accessible.

A distinction is made between absolute and relative URIs.

Absolute URIs have the following general structure:

<schema> : <schema-specific part>

The **schema** defines the namespace of the URI as well as the syntax and meaning of the schema-specific part. A schema is usually specified by the access protocol. However, this does not always have to be the case.

The following examples show some common URIs:



example

Some common URIs

- Schema for HTTP services
`http://www.oncampus.de/oncampus-gmbh.html`
- Schema for mail services
`mailto:firstname.surname@fh-luebeck.de`
- Schema for FTP services (the URI syntax is described in RFC 2396)
`ftp://ftp.xyz.com/rfc/rfc2396.txt`
- Schema for accessing local files
`file://E:/services/index.html`
- Schema for Telnet services
`telnet://abc.oncampus.de`

The syntax of the schema-specific part is in general not defined, but hierarchical structures are often used. These URIs are composed of the following four parts:

`<Schema>://<authority><path>?<request>`

Before the authority part there is: `“//”`. It is terminated by `“/”`, question mark `“?”` or the end of the character string. Additional parameters can be listed in the request section. The authority part is constructed as follows:

`<authority> = <userinfo>@<host>:<port>`

Often, the format can be used to identify a user in the userinfo part **“User: password”** – but it must be kept in mind that these details are entered in the address line of the browser in a way that is readable for everyone and transmitted in plaintext; they therefore always involve a security risk. The host part can have a DNS name or an IP address. A port number can be specified. For each resource, fragments separated by `“#”` from the URI can be specified – in HTML pages, these are the **local links** within a page.

Relative URIs do not begin with either a schema (protocol) or a slash `“/”`. The entries `“.”` and `“..”` have a special meaning for relative references, namely the `“current hierarchy level”` or a `“level above this hierarchy level.”`



indentation

Absolute and relative URIs

Absolute references


- <http://www.oncampus.de:80/index.html?p1=new&p2=123>
- <telnet://193.170.110.12:23>
- <ftp://meier:reiem@193.170.110.12:21/v1/dat.zip>
- <http://www.vfh.de/index.html#a1>

Relative references

Relative references are typically used only within a site. This allows a whole tree of resources to be moved without having to change the references. This is, for example, important when a fully prepared site is supposed to be transferred to the provider for publication. Some examples for this are:


- [./pictures/picture1.jpg](#)
- [../pictures/picture2.gif](#)
- [index.html](#)

6.2.2 Hypertext Markup Language

The Hypertext Markup Language (HTML) will only briefly be discussed here in order to classify it. There are many sources on the Internet on this language (see e.g. [W3 Schools](#) )

HTML is a language specification that defines a structure in addition to the content of websites. For this structure, so-called **tags** are used, e.g. the tag `h1` for a heading. Opening and closing tags result in a logical tree structure (example: `<tr><tr>` for tables, with `tr` for a table row, and `td` for a table cell). A main feature of HTML are the hyperlinks, which can be used to refer to other locations in the same document but also to other documents. Other documents can be found on the same web server, but can also be found in completely different places on the Internet.

At the beginning, HTML documents contained instructions on how they should be displayed by the browser (e.g. which colors should be used). However, these display instructions were then outsourced to special documents, the **cascading style sheets**. This has the great advantage that different HTML files can use a CSS file thereby creating a uniform web site.

From 1992 to 1999, the presentation possibilities of HTML were extended over several versions, up to the version 4.01. After that, the development was not continued for a long time by W3C; this led to the development of proprietary solutions for the realization of additional features (in particular Adobe Flash). After the work was resumed within W3C, **HTML5**  was defined in October 2014. This language version quickly spread widely.

While at the beginning of the WWW, HTML documents were written with editors, tools for the creation of the pages developed relatively soon. Today's **content management systems** (e.g. [Wordpress](#)) have evolved on this basis.



websource

Here are some additional links related to HTML.

A [validation service](#) is offered by W3, which can be used to check HTML pages for compatibility with the standard. If you want to make websites accessible, you can also use the [HTML code sniffer](#) to check compliance with the [WCAG guidelines](#). The color selection in the web page can be checked with the [ContrastA](#) tool to see if a sufficiently large contrast is used.

6.2.3 Tasks of Client and Server

For the retrieval of web pages, a **browser** is used on the user side, which can also be referred to as a **WWW client**. After a URL has been entered into the browser line, the browser first accesses the DNS to determine the appropriate IP address for the domain name. After this, the page is requested from the server, which in many cases nowadays consists of an HTML framework and references to objects (e.g. images), CSS and JavaScript files (and possibly other active content such as ActiveX). The browser also retrieves these files after evaluating the HTML code. The instructions in the CSS files are processed accordingly, and the active contents are executed as well. To display some content such as pdf files, the browser uses plug-ins.

The protocol **HTTP** is used for communication between the browser and a web server. The **web server** receives HTTP requests and evaluates the URL for the desired object. It reads the object from the file system and transfers the object in an HTTP response. The basic principle is that a web server is stateless, i.e. each request/response is considered independently of previous requests/responses. This principle can, however, be reversed by HTTP cookies, which is almost always done for today's websites.



indentation

Browser market

While Netscape Navigator was the most widely used browser in the 1990s, Internet Explorer dominated the market for several years afterwards because it was delivered together with the Windows operating system. In the last few years, the Google Chrome

browser has gained more and more market share, so it now has the most users (see [browser statistics](#)). However, the measurement of the market share of browsers is subject to some uncertainties.

6.2.4 HTTP



arrangement

6.2.4 HTTP

6.2.4.1 [Functionality Overview](#)

6.2.4.2 [Non-persistent HTTP](#)

6.2.4.3 [Multiple Simultaneous Connections](#)

6.2.4.4 [Persistent HTTP](#)

6.2.4.5 [HTTP Example](#)

6.2.4.6 [Provisioning of Today's Web Sites](#)

6.2.4.7 [Request Methods](#)

6.2.4.8 [Status Messages](#)

6.2.4.9 [Caching](#)

As mentioned before, **HTTP** (Hypertext Transfer Protocol) is used for retrieving WWW pages.

The first protocol version 0.9 is from 1991. It was relatively simple and was not published as an RFC. In 1996, version 1.0 was defined as RFC 1945; version 1.1 (RFC 2616 from 1999) brought some improvements, especially keep-alive.

After the development had not been continued for many years, HTTP/1.1 was again specified in RFCs 7230 to 7235 in 2014. There were no changes to the content since only the presentation of the specification was supposed to be improved.

In 2015, however, the version HTTP/2 (RFC 7540 and RFC 7541) was issued, which brought some significant changes in order to achieve higher performance.



In the online version an video is shown here.

Link to video : http://www.youtube.com/embed/z_cWVHE2_9c

HTTP

6.2.4.1 Functionality Overview

HTTP is a **stateless request/response protocol**. A request is sent from client to server specifying the request method, URI, and protocol version, followed by details for modifying the request. Client information and possibly content can follow.

The server responds with a status message followed by information about the server, meta information about the content, and possibly also content.

In its basic version, HTTP does not know any states so that each new request/reponse is independent from previous requests. This means it is not possible without special arrangements to define sessions as required for applications with shopping carts, e.g. for orders in a bookshop. In RFC 6265, an extension for HTTP is specified in order to handle states by exchanging **cookies** between server and client. Cookies are identifiers that are stored within client and server and which the client provides as part of further requests. This allows the server to recognize that it has previously communicated with this client.



annotation

The cookies that the Firefox browser currently has stored can be viewed by entering “about:preferences#privacy” in the browser line.

6.2.4.2 Non-persistent HTTP

The first web pages contained a lot of text and only a few images, so that the load time was not too long. Therefore, it was not a big problem at first that the original method of web page retrieval was relatively inefficient.


This method, which is called **non-persistent HTTP** and is part of the versions HTTP/0.9 and HTTP/1.0, first opens a TCP connection to access the web page. This provides an HTML file to the client that can contain references to objects contained in the web page. For example, an image is not retrieved directly, and the reference to the image is obtained first. After downloading the HTML code, the TCP connection is closed.

Afterwards, TCP connections are opened and closed **sequentially** to retrieve the objects. Only one object is thereby retrieved per TCP connection. Thus, first a TCP connection is opened, the first object is requested and received, and then the TCP connection is closed. After this, another TCP connection is used for the second object, and so on.

The method is unfavorable because many TCP segments are required for the repeated connection setups and closings. In addition, the bit rate within the TCP connections remains low because they exist for only a short time (remember Slow Start).



annotation

Some webpages can still be retrieved using the original method, which is also possible independent from a browser. For example, you can type in the command line: "telnet google.com 80". After the connection is set up, enter: "GET /about/" to get the Google page as text. Remark: For Windows it is necessary to enable the telnet client (see [instructions](#) ) . This makes sense because telnet is insecure and should not be not used regularly.

6.2.4.3 Multiple Simultaneous Connections

An improvement of the original method is to perform queries of objects in parallel (**multiple simultaneous connections**). At the beginning, there is no difference when the HTML page is accessed at first. Afterwards, however, several TCP connections (e.g. six or eight) are opened in parallel, with an object being requested in each TCP connection. The number of TCP connections remains unchanged compared to the previous method. If the webpage contains many objects, then the specified number of TCP connections is set up several times. The query becomes more efficient because waiting times can be used to set up additional connections. It is also possible to obtain a higher proportion of the total bit rate with several simultaneous TCP connections (see [Fairness](#)).

6.2.4.4 Persistent HTTP

Persistent connections were introduced in extensions of HTTP 1.0 and then as recommended method in HTTP 1.1. They are also known as **keep-alive**. The idea is that the entire page is retrieved via a single TCP connection. This TCP connection is therefore not closed after the HTML framework for the webpage has been retrieved; instead, object retrievals are also made via this TCP connection.

Persistent connections have several advantages:

- Because there are fewer TCP connections, fewer resources such as CPU time and buffers are required in hosts, routers, and proxies, since frequent TCP connection setups and closings are no longer required.
- The available bit rates capacities can be better used because the TCP "slow start" phase is less frequent, and TCP can adapt better to the bit rate capacity in a longer-lasting connection. Overload situations are reduced because fewer packets are sent.

The method can be made even more efficient by executing several requests for objects in parallel within the TCP connection. This is called **request pipelining**; it can be viewed as a combination of persistent connections with multiple simultaneous connections. However, the efficiency improvement can only be slight if the requests are processed sequentially on the server side.

Persistent connections are usually terminated if a connection has been inactive for a certain time, e.g. 60 seconds. If the server terminates the connection and makes a new request exactly at the time the client sends a new request, then a new, second connection must be established. However, no more than two concurrent connections should be used.

In the way as we have considered persistent connections so far, they only offer advantages. Unfortunately, however, there is a possibility of attacks by malicious clients since this method requires the client to initiate connection decommissioning. This is because the server does not know how many objects the client will request in the following. Malicious clients can open many TCP connections that are closed only after relatively long timeouts so that a maximum number of possible TCP connections is reached. Normal users can then no longer reach the web page (**denial of service attack**). The problem also cannot be completely eliminated by a significant reduction of the timeout value, e.g. to 2 seconds.

6.2.4.5 HTTP Example

The following trace shows some mechanisms of HTTP. The start page is requested without being specified exactly. Only the IP address and the directory /test are given. The start page that is transmitted depends on the server. Often, it is the index.html page. A style sheet file format.css is referenced by the start page; the style sheet must also be transmitted. The start page is requested again after 81 seconds.

In the following interactive graphic, you can learn more about the mechanisms of HTTP by hovering over the line numbers in the first column.



In the online version an rollover element is shown here.

Trace: Example of a communication via HTTP

Begin printversion

No	Time	Source	Destination	Protocol	Info
1	0.00	192.168.0.1	192.168.0.2	TCP	1096 > 80 [SYN]
2	0.00	192.168.0.2	192.168.0.1	TCP	80 > 1096 [SYN, ACK]
3	0.00	192.168.0.1	192.168.0.2	TCP	1096 > 80 [ACK]
4	0.00	192.168.0.1	192.168.0.2	HTTP	GET /test/ HTTP/1.1
5	0.01	192.168.0.2	192.168.0.1	HTTP	HTTP/1.1 200 OK
6	0.20	192.168.0.1	192.168.0.2	TCP	1096 > 80 [ACK]
7	0.20	192.168.0.2	192.168.0.1	HTTP	Continuation
8	0.22	192.168.0.1	192.168.0.2	HTTP	GET /test/format.css HTTP/1.1
9	0.22	192.168.0.2	192.168.0.1	HTTP	HTTP/1.1 200 OK
10	0.23	192.168.0.2	192.168.0.1	HTTP	Continuation
11	0.23	192.168.0.1	192.168.0.2	TCP	1096 > 80 [ACK]
12	0.23	192.168.0.2	192.168.0.1	HTTP	Continuation
13	0.40	192.168.0.1	192.168.0.2	TCP	1096 > 80 [ACK]
14	60.73	192.168.0.1	192.168.0.2	TCP	1096 > 80 [RST]
15	81.21	192.168.0.1	192.168.0.2	TCP	1097 > 80 [SYN]
16	81.21	192.168.0.2	192.168.0.1	TCP	80 > 1097 [SYN, ACK]
17	81.21	192.168.0.1	192.168.0.2	TCP	1097 > 80 [ACK]
18	81.21	192.168.0.1	192.168.0.2	HTTP	GET /test/ HTTP/1.1
19	81.23	192.168.0.2	192.168.0.1	HTTP	HTTP/1.1 304 Not Modified
20	81.23	192.168.0.1	192.168.0.2	HTTP	GET /test/format.css HTTP/1.1
21	81.23	192.168.0.2	192.168.0.1	HTTP	HTTP/1.1 304 Not Modified
22	81.40	192.168.0.1	192.168.0.2	TCP	1097 > 80 [ACK]
23	141.74	192.168.0.1	192.168.0.2	TCP	1097 > 80 [RST]

<i>field</i>	<i>description</i>
1 - 3	A TCP connection to the server using its port 80 is established.
4	Request of the start page in the directory /test/.
5	Reply from the WWW (world wide web) server. Just header information is sent.
7	Reply from the WWW server. The requested page is transferred.
8	Request for the page /test/format.css.
9	Reply from the WWW server. Just header information is sent.
10 and 12	The requested page is transferred. Because it is too large, it is delivered in two parts.
14	The TCP connection is terminated after 60 seconds with a reset.
15 - 17	On page renewal after 81 seconds a new TCP connection is set up.
19 and 21	The pages are not provided again by the WWW server because they have not been modified. The browser shows the pages from its local cache.
23	The TCP connection is terminated by a reset after not being used for further 60 seconds.

End printversion

The HTTP requests and responses are displayed in the trace. Each request and response has several parameters, which are shown below.

- 1 to 3: A TCP connection to the server, more precisely to server port 80, is set up.
- 4: Request the start page in directory /test/.
- 5: Response from the WWW server. Only header information is sent.
- 7: Response from the WWW server. The requested page is transferred.
- 8: Request of the page /test/format.css.
- 9: Response from the WWW server. Only header information is sent.
- 10 and 12: The requested page is transferred. Since it is too large, it is sent in two parts.
- 14: After 60 seconds, the TCP connection is terminated with a reset.
- 15 to 17: A new TCP connection is established to update the pages after 81 seconds.
- 19 and 21: The pages are not again transferred from the WWW server because they have not been modified. The browser displays the pages from its local cache.
- 23: After the TCP connection has not been used for another 60 seconds, it is terminated by reset.

There are two important principles:

- The TCP connection is not terminated after every transferred file but only after it has not been used for a certain time (persistent connection).
- Files which are requested multiple times are only transferred again if they have been changed.

Let us look at some of the segments in more detail:

The start page is requested and transmitted in the segments 4, 5 and 7:

Segment 4

Hypertext Transfer Protocol

```
GET /test/ HTTP/1.1\r\n
Accept: */*\r\n
Accept-Language: de\r\n
Accept-Encoding: gzip, deflate\r\n
User-Agent: Mozilla/4.0\r\n
Host: 192.168.0.2\r\n
Connection: Keep-Alive\r\n
```

The start page in the /test/ directory is requested. Connection keep-alive specifies that the connection should not be terminated after transmission of the page.

Segment 5

Hypertext Transfer Protocol

```

HTTP / 1.1 200 OK \ r \ n
Server: Microsoft-IIS/4.0\r\n
Content-Location: http://192.168.0.2/test/index.html\r\n
Date: Fri, 21 Jun 2002 8:53:36 AM GMT\r\n
Content-Type: text/html\r\n
Accept-Ranges: bytes\r\n
Last-Modified: Fri, 21 Jun 2002 8:27:28 AM GMT\r\n
ETag: "0d8737afd18c21:5ef6"\r\n
Content-Length: 225\r\n
\r\n

```

A start page has been found, which is indicated by the response code 200. Content-location displays the exact location of the start page. Content-length specifies the length of the start page. However, the start page itself is transmitted in the next segment. Last-modified and Etag (entity tag) convey data that are sent back to the server when the page is requested again. This allows the server to determine whether the page has changed. If these values are unchanged, the page is not retransmitted.

Segment 7

Hypertext Transfer Protocol

\r\n

Data (225 bytes)

0000	3c 21 44 4f 43 54 59 50 45 20 48 54 4d 4c 20 50	<!DOCTYPE HTML P
0010	55 42 4c 49 43 20 22 2d 2f 2f 57 33 43 2f 2f 44	UBLIC "-//W3C//D
0020	54 44 20 48 54 4d 4c 20 34 2e 30 20 54 72 61 6e	TD HTML 4.0 Tran
0030	73 69 74 69 6f 6e 61 6c 2f 2f 45 4e 22 3e 0d 0a	sitional//EN">..
0040	3c 6c 69 6e 6b 20 72 65 6c 3d 73 74 79 6c 65 73	<link rel=styles
0050	68 65 65 74 20 6d 65 64 69 61 3d 22 73 63 72 65	heet media="scre
0060	65 6e 22 20 74 79 70 65 3d 22 74 65 78 74 2f 63	en" type="text/c

0070	73 73 22 20 68 72 65 66 3d 22 66 6f 72 6d 61 74	ss" href="format
0080	2e 63 73 73 22 3e 0d 0a 3c 68 74 6d 6c 3e 0d 0a	.css">...
00e0	0a	

This is the requested HTML page with the reference to the format.css file.

The file format.css is transferred in the segments 9, 10 and 12:

Segment 9

Hypertext Transfer Protocol <br

sp; sp; HTTP / 1.1 200 OK \ r \ n

Server: Microsoft-IIS/4.0\r\n

Date: Fri, 21 Jun 2002 08:53:36 GMT\r\n

Content-Type: text/css\r\n

Accept-Ranges: bytes\r\n

Last-Modified: Fri, 21 Jun 2002 08:28:02 GMT\r\n

ETag: "0d5b78efd18c21:5ef6"\r\n

Content-Length: 2375\r\n \r\n

Content-location does not exist because the name of the file was specified in the request.

Segment 10

Hypertext Transfer Protocol

\r\n

Data (1458 bytes)

0000 09 62 6f 64 79 2c 68 31 2c 68 32 2c 68 33 2c 68 .body,h1,h2,h3,h

... and so on

This is the first part of the format.css file. Because the content-length in segment 9 is greater than the maximum length of an Ethernet frame, the requested file is sent in two parts; HTTP determines how large the parts are. HTTP must therefore consider the maximum segment size (MSS), which was exchanged during the TCP connection setup. The packet therefore does not need to be fragmented by IP; instead, "don't fragment" is set in the IP header.

Segment 12

 Hypertext Transfer Protocol

\r\n

Data (901 bytes)

... and so on

This is the second part of the format.css file.

In the segments 18 and 19, the start page is requested again after a new TCP connection has been established:

Segment 18

Hypertext Transfer Protocol

GET /test/ HTTP/1.1\r\n

Accept: */*\r\n

Accept-Language: de\r\n

Accept-Encoding: gzip, deflate\r\n

If-Modified-Since: Fri, 21 Jun 2002 08:27:28 GMT\r\n

If-None-Match: "0d8737afd18c21:5ef6"\r\n

User-Agent: Mozilla/4.0\r\n

Host: 192.168.0.2\r\n

Connection: Keep-Alive\r\n

\r\n

If-modified-since is the date of the last change of the page that resides in the local cache. If the cache is cleared in the meantime, this is missing. If-non-match is the value for the Etag that was delivered by the server when the page was requested. If one of these two items has changed in the meantime, the page is retransmitted; otherwise, the page is displayed from the local cache of the client system.

Segment 19

Hypertext Transfer Protocol

/1.1 304 Not Modified\r\n

Server: Microsoft-IIS/4.0\r\n

Date: Fri, 21 Jun 2002 8:54:57 AM GMT\r\n

Content-Location: http://192.168.0.2/test/index.html\r\n



ETag: "0d8737afd18c21:5ef6"\r\n

\r\n

The page has not been modified and will therefore not be transmitted.

6.2.4.6 Provisioning of Today's Web Sites

In order to have a better idea of the use of the fundamental methods in practice, a study from a bachelor's thesis at the FH Lübeck (Q. Zhang, WS 2015/16) is discussed here. It can be regarded as an update of a study from a paper from 2009 (Charzindski, DFN-Forum 2009).

The student examined the access to the world's most popular 50 websites according to the [Alexa ranking](#) . The following values were measured using the test page webpagetest.org .

- Data transferred
- Number of loaded elements (using HTTP GET)
- Number of different servers to which connections are established
- Number of different network prefixes (NP) and autonomous systems (AS)
- Number of different “second level domains” in DNS



Measured values for the delivery of the 50 most popular homepages worldwide

Begin printversion

	KBytes	HTTP Requests	Servers	AS	DNS SLDs
Minimum	20	1	2	1	1
Average	1814.2	94.42	12.32	2.5	4.14
Maximum	11,447	508	36	4	16

End printversion

It is thus shown that if an average homepage is loaded, the following can be observed.

- About 94 HTTP requests are necessary, i.e. there are a lot of objects contained.
- Twelve different servers, which are located in more than two autonomous systems, are contacted. This means that a website is no longer just delivered by a single server, which was the assumption in the previous considerations.
- “Helper services,” especially akamai.net and google.com, are often used.

6.2.4.7 Request Methods

HTTP defines different request methods that can be used in a request. The methods are briefly presented below.

For each request, the method, the URI, and the HTTP version are specified in the first line. The request can be specified more precisely in the subsequent header lines. Some methods allow that also data can be sent.

GET

GET is used to request information from the server that is uniquely identified by its URI. This can be an HTML page for example. Conditions can be given as part of a GET request. This allows to reduce the network load and not retransmit pages that are already in the client's cache or proxy servers. These conditions are specified in the header fields *if-modified-since*, *if-unmodified-since*, *if-match*, *if-none-match*, or *if-range*. In addition, it is possible to transfer only parts of a file.

HEAD

HEAD leads to the same result as GET, with the only difference that the actual contents are not transmitted. This allows to check the existence of links. It can also be determined whether pages can be accessed or have been modified.

POST

With POST, data can be transferred to the servers that are related to the specified URI. The resource identified by the URI is responsible for how the data is to be processed. POST is intended for the following applications:

- Transfer of data from a form for further processing, e.g. for storing in a database.
- To post a message, for example, to a news group.
- To add comments to an existing resource.

PUT

PUT is used to send data to the server that will overwrite or create the specified URI if the resource is not already present.

DELETE

DELETE is used to delete the resource identified by the specified URI.

CONNECT

CONNECT is only used in conjunction with a proxy server, e.g. to establish a secure connection using SSL (Secure Sockets Layer).

OPTIONS

With OPTIONS, the communication methods can be queried in the request-response chain of the resource identified by the specified URI.

TRACE

TRACE activates a loopback to the application level. The receiver of a message sends it back to the sender. This method is used for testing or diagnosis.



example

In the following example, data from a form is transmitted to the server with the GET and alternatively with the POST method.

With GET, the data are sent as a request – separated from the URI by “?”.

```
<span style="font-family: Courier New">Hypertext Transfer Protocol
GET /test/index1.html?feld=xx4&submit=Request+send HTTP/1.1\r\n
Accept: */*\r\n
Referrer: http://192.168.0.2/test/index1.html
<span style="font-family: Courier New"> Accept-Language: de\r\n
Accept-Encoding: gzip,
deflate\r\n User-Agent:
Mozilla/4.0\r\n Host:
192.168.0.2\r\n Connection:
Keep-Alive\r\n
\r\n <span>
```

With POST, the data from the form is sent as a data block following the header fields.

```
<span style="font-family: Courier New"> Hypertext Transfer Protocol
POST /test/index4.html HTTP/1.1\r\n
Accept: */*\r\n
Referrer: http://192.168.0.2/test/index.html
<span style="font-family: Courier New"> Accept-Language: de\r\n
Content type: application/x-www-form-urlencoded\r\n
Accept-Encoding: gzip, deflate\r\n
User-Agent: Mozilla/4.0\r\n
Host: 192.168.0.2\r\n
Content-Length: 30\r\n
Connection: Keep-Alive\r\n
Cache-Control: no cache\r\n
\r\n
Data (30 bytes)
```

```
0000 66 65 6c 64 3d 78 78 34 26 73 75 62 6d 69 74 3d field=xx4&submit=
0010 41 6e 66 72 61 67 65 2b 73 65 6e 64 65 6e Request+send
```

6.2.4.8 Status Messages

If the server responds, the first line displays the HTTP version and a status message. Additional information can be provided in the subsequent header lines. Some methods allow to send data as well.

Like other Internet protocols, HTTP also recognizes different classes of status messages that are sent to the client in the responses from the server:

1xx	Provisional response, e.g. 100 Continue.
2xx	Successful request, e.g. 200 OK.
3xx	Redirection, e.g. 301 Moved Permanently, 304 Not Modified.
4xx	Client error, e.g. 404 Not Found.
5xx	Server error, e.g. 500 Internal Server Error.



Classes of status messages

6.2.4.9 Caching

In practice, many pages are retrieved from the WWW. Therefore, the question arises of whether the delivery can be made more efficient if many users are interested in the same content.

First of all, each browser has its own cache for pages that the user has already viewed. For example, in Firefox, you can view the cache contents by entering “about:cache” into the browser line. In addition, it would be possible for an organization to run a central web caching in order to minimize the Internet connection traffic. This was done in the past, for example, by many universities, but it is no longer important today. On the one hand, there are now considerably larger bit rates available for the network connectivity, and on the other hand, with this type of caching, one cannot really estimate very well whether websites might have changed.

The caching therefore takes place today at the information providers, which often rely on specialized service providers, so-called **content distribution networks** (CDNs), for this purpose (see, for example, [the website of the CDN provider Akamai](#)). The CDNs consist of many servers that contain duplicates of web pages. If you retrieve a web page as a user, it is delivered by the server that is the closest in order to achieve a low latency. The provider can ensure that the distributed content is up-to-date.

In practice, an information provider can use CDNs only partially. The CDN can, for example, distribute static parts of the page with large data volumes (e.g. title images), while the actual textual information is delivered directly by the provider.

6.2.5 HTTP/2

In May 2015, a new version of HTTP called HTTP/2 was specified in RFC 7540, which is backward compatible with the previous versions. The development was largely influenced by Google's SPDY proposal. HTTP/2 was accepted very quickly by browsers and servers, as can be seen in the statistics from the CDN provider [KeyCDN](#). You can also see there whether a web page is already delivered with HTTP/2 (see [test page](#)). This is not recognizable in the browser line.



In the online version an video is shown here.


Link to video : <http://www.youtube.com/embed/Krib5I8ObbI>

HTTP/2

The faster delivery of websites was in the focus of the HTTP/2 introduction (see [HTTP/2 page of Akamai](#)). It is achieved through the following measures.


- Similar to the persistent connections, only one TCP connection exists. Within the TCP connection, different streams exist for parallelizing the data transmission.
- The single data unit used for transmission is called a frame (not to be confused with the layer 2 data units). A frame can contain data from several streams so that small data units are avoided.
- It is possible to prioritize the contents of web pages so that important parts are delivered first. The highest priority is given to the HTML pages. This is followed by the CSS files and then the JavaScript files. Images have the lowest priority.
- It is also possible that the server delivers pages or objects that are not requested by the client. This is useful if the server side has experience regarding which requests are typically made by the client in which order.

- In HTTP/1.1 and also in the previous versions, the header consists of fields in plaintext. To make this more efficient, a compression method called HPACK was defined for the header (RFC 7541).

The initial discussion upon the introduction of HTTP/2 considered whether a combination with TLS encryption was to be declared mandatory. But this was ultimately not specified in this way. Leading browser vendors (Chrome, Firefox, etc.), however, only support encrypted connections (see [HTTP/2 FAQ](#) .



notice



A [book](#)  (“High Performance Browser Networking” by Ilya Grigorik) on new developments in this area, especially on HTTP 2.0, is available free of charge at O'Reilly.

6.3 E-Mail



arrangement

- 6.3 [E-Mail](#)
- 6.3.1 [SMTP Basics](#)
- 6.3.2 [SMTP Operation](#)
- 6.3.3 [MIME](#)
- 6.3.4 [Security](#)
- 6.3.5 [Mail Retrieval](#)

In addition to the WWW, electronic mail is probably the most used Internet service today, with more than 250 billion e-mails sent per day (see [statista.com](#) ). It goes back to [Ray Tomlinson](#) , who developed it in 1971 and also suggested the use of the @ character (it was previously used for prices, e.g. “five items @ \$1.95”). He combined the transmission of files over networks with existing message systems, which users of a main frame could send messages to each other.



In the online version an video is shown here.

Link to video : <http://www.youtube.com/embed/8LPntEmoxlc> 

E-Mail

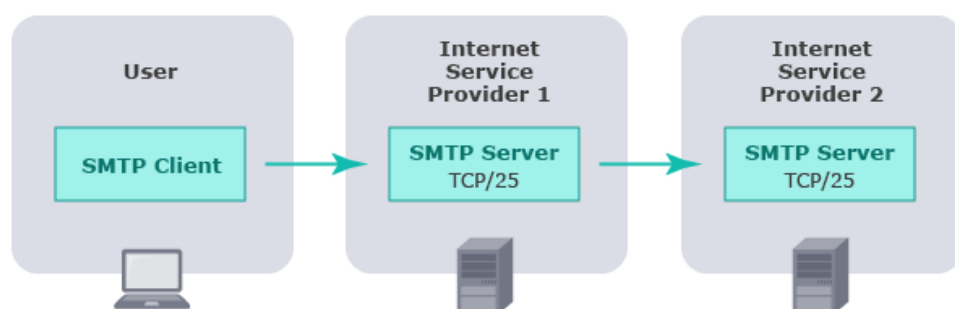
Many terms are taken from the traditional postal service: There is **mail** with **envelope** and **content**, **mailboxes**, recipient as well as sender addresses. It is also very similar to the postal service in terms of delivery. The sender sends mail through the post office (the mail system), which carries it through various intermediate stations (mail servers), although it is still not at all clear at this point whether the receiver exists and has a mailbox. As a rule, the sender does not receive any information as to when the letter was inserted into the mailbox of the recipient and whether the letter was read by the recipient – except if the recipient returns a reply letter.

Sending an electronic mail is a relatively simple process: You only need to know a mail server to which the e-mail can be sent. This mail server ensures that another mail server is addressed – if it is not already have access to the mailbox of the recipient and can deliver the e-mail to the mailbox. The communication from the sender of the email to the mail server as well as between the different mail servers is carried out using **SMTP (Simple Mail Transfer Protocol)** (first published in RFC 821, currently RFC 5321). SMTP has mainly prevailed because of its simple structure.



annotation

The development of different protocols has shown that simple protocols (such as SMTP) with few parameters are used a lot, while complicated protocols are often unclear and lead to difficulties in communication.



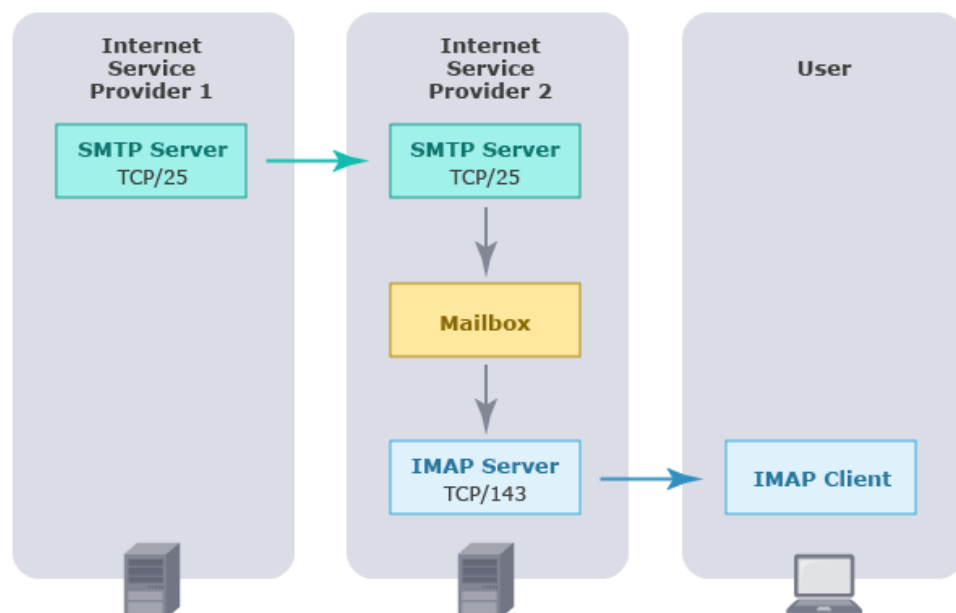
Sending e-mails

The arrows indicate the direction in which e-mails are sent. No content of e-mails is sent in the opposite direction of the arrows; only information about an e-mail, confirmations or error messages are sent in this direction.

Receiving e-mail is more complicated. For this purpose, a mailbox must exist, i.e. a mail server must be in operation, which stores the received e-mails in the mailbox of the recipient. This computer is generally not operated by the recipient itself because if the

recipient turns off its computer the mailbox is no longer reachable. In order to have an uninterrupted mailbox availability, the corresponding host must run 24/7. Therefore, the mailbox is in many cases located on the mail server of the Internet service provider, via which the user is connected to the Internet. The user can access its mailbox at the ISP and read the e-mails if necessary. Certainly, other mail services located somewhere else in the Internet can also be used.

The communication between the user and the mail server for reading the e-mails is carried out today with IMAP4 (**I**nternet **M**essage **A**ccess **P**rotocol - version 4, RFC 3501) or via WWW browser (so-called **w**eb **m**ail). The older POP3 (**p**ost **o**ffice **p**rotocol - version 3) is only rarely used.



Receiving e-mails with IMAP

The arrows indicate the direction in which e-mails are sent. No content of e-mails is sent in the opposite direction of the arrows; only information about an e-mail, confirmations or error messages are sent in this direction.

Mail systems for the user consist of an SMTP client for writing and sending e-mails and a POP3 and/or IMAP4 client for reading and managing the received e-mails or a web browser that includes an SMTP client.

All e-mail protocols are based on TCP to take advantage of its reliability because ultimately the e-mails need to arrive completely and correctly.

6.3.1 SMTP Basics

Mail objects are transferred via SMTP. A mail object consists of an envelope and the content.

The **envelope** consists of the producer address (SMTP command: MAIL FROM), one or more recipient addresses (SMTP command: RCPT TO) and optionally of additional data. Only the 7-bit US-ASCII code can be used for information in the envelope. The highest bit in each byte is set to 0. For example, umlauts (German letters ä,ö,ü) are not possible with this code.

The **content** (SMTP command: DATA) consists of the headers (all in the format name:value) and the body. The header and body consist of text and also use the 7-bit US-ASCII code. 8-bit values can be sent in the body only with an extension of the SMTP protocol ("8BITMIME," see [MIME](#)). In the headers, however, still only the US-ASCII code is permitted with MIME; there is, however, an extension for the use of umlauts.

A **mail address** identifies a user to whom the mail is directed or a **mailbox** in which the mails are stored. Both terms are often used synonymously. A standard mail address consists of a local part, which can be much more than a user name, and a domain specification: **local-Part@domain**. The local part may only be interpreted by the host to which the domain refers – no relay or gateway SMTP host (see below) may interpret or even modify this part. The local part of a mail address is case-sensitive, but not the domain: `firstname.name@fh-luebeck.de` can be a different mail address than `Firstname.Name@fh-luebeck.de`! In order to eliminate difficulties of this kind right from the start, the local part should always be written in lower case.



example

The following example shows information that was sent in addition to the actual body of an e-mail:

```
<span style="font-family: Courier New"><span style="color:green;">Received: from
dns.fh-luebeck.de by ru.fh-luebeck.de with SMTP
(1.37.109.18/16.2) id AA139362930; Mon, 10 Nov 1997 12:55:31 +0100
Return-Path: <testuser@example.com>
Received: from mrin39.mx.example.com by dns.fh-luebeck.de with SMTP
(1.37.109.4/16.2) id AA12255; Mon, 10 Nov 97 12:39:57 +0100
Received: (from root@localhost)
  by mrin39.mail.example.com (8.8.5/8.7.3/AOL-2.0.0) id GAA09416 for
firstname.name@fh-luebeck.de;
```



```
Mon, 10 Nov 1997 06:41:37 -0500 (EST)
<span style="font-family: Courier New"><span style="color:blue;">Date: Mon, 10
Nov 1997 06:41:37 -0500 (EST)
From: testuser@example.com
Message-Id: <971110064134_1079671197@mrin39>
To: firstname.name@fh-luebeck.de
Subject: Testmail
Mime-Version: 1.0 boundary="PART.BOUNDARY.0.895.mrin39.879162093"
Status: U
X-PMFLAGS: 570949760 0
Content-ID: <0_895_879162094@mrin39.2214>
Content-type: text/plain
<span style="font-family: Courier New"><span style="color:red;">Dear Mr. XY,
this is a test mail.

The producer system, a relay system, and the delivery system have added trace
information and a return path (green). The mail contains different header information
(blue). Finally, the actual content (red) follows.
```

When transmitting e-mails, a distinction can be made between different types of **SMTP systems** according to their function:

- The **producer** system sends the e-mail to the Internet or, more precisely, to a transport system.
- The **delivery** system receives the e-mail from the Internet or – more precisely – from a transport system and stores it in a mailbox.
- A **relay** system receives the e-mail and transmits it. The content of the e-mail is not changed. However, trace information is added: time-stamp information (**received**) and possibly an e-mail address to be notified in case of any errors (**return path**).
- A **gateway** system receives the e-mail and also modifies its contents, e.g. due to incompatibilities between different transport systems. Firewalls that rewrite addresses can also be viewed as gateways.

6.3.2 SMTP Operation

The e-mail sender uses SMTP commands to which the e-mail recipient responds. All replies begin with a 3-digit number code.

**SMTP Commands**

Begin printversion

SMTP command	Explanation
HELO, EHLO	Identification of the client towards the server
MAIL FROM	Beginning of mail transaction: mail sender
RCPT TO	Mail recipient
DATA	E-mail contents
RSET	Reset of a mail transaction
VERFY	Verification of a user or mailbox, respectively
EXPN	Expansion of receiver list
HELP	Sending of help information
NOOP	No operation except of sending a positive acknowledgment
QUIT	Quit and terminate connection

End printversion

Reply to commands

The three digits in a reply have special meanings. The first number indicates whether the request was successful, incompletely processed or not processed at all. The approximate kind of error is encoded in the second digit. The information of the second digit is described in more detail with the third digit.

**General meaning of the replies**

Begin printversion

Sequence of digits	Explanation
1yz	Positive, preliminary reply
2yz	The request has been successfully processed and answered.
3yz	Positive intermediate reply
4yz	The command has preliminary not been accepted and not processed. The command can be sent again.
5yz	The command has finally not been accepted.
x0z	Related to syntax, e.g. syntax error
x1z	Related to information, e.g. status or help
x2z	Related to connections
x5z	Related to mail system

End printversion

The most important replies are:



The most important replies

Begin printversion

Sequence of digits	Explanation
220	Service ok.
221	Service terminates transmission.
421	Service is not available.
250	Requested action is ok.
354	Request to enter an e-mail, end it via <CRLF>.<CRLF>.
450	Requested action cannot be performed, e.g. mailbox is locked.
550	Requested action cannot be performed, e.g. mailbox not found.
552	Mail action aborted: e.g. out of memory.
553	Mail action not executed: e.g. wrong mailbox syntax.
500	Syntax error, command not recognized.
501	Syntax error in parameter.
502	Command not implemented.
503	Sequence of commands is wrong.
504	Command parameter not implemented.

End printversion



EHLO

Today ESMTP (Extended SMTP) is used almost exclusively. This makes it possible to define additional commands or keywords, which must be registered at IANA.

In an ESMTP session, several phases are passed through:

- The client establishes a TCP connection to port 25 of the server.
- The server then sends a welcome response: 220...
- The client now sends **EHLO**, whereupon the server responds with 250 The server can send multiple 250 replies with fixed keywords, such as



Examples of 250-replies

Begin printversion

250-replies	Explanation
250 8BITMIME	Data can be sent in the 8 bit code. For doing so, the key word BODY is given in addition to the command MAIL FROM. BODY gets an argument that indicates the kind of encoding.
250 SIZE	The maximum size of an e-mail to be sent can be requested from the server. This is interesting if large mails should be sent.
250 HELP	There is a help function. Since this command is optional, it is explicitly listed if it is supported.

End printversion

You can also use privately agreed keywords in addition those registered at IANA; they must start with "X".

- The client sends MAIL FROM with its sender address and receives the reply 250 Ok.
- The client sends RCPT TO with its recipient address and receives the reply 250 Ok. These two items form the envelope for the e-mail; the envelope is used to deliver the e-mail to the mail server that manages the mailbox of the recipient.
- The client sends DATA and receives the reply 354 ... which states that the server expects the contents of the e-mail. Now, different headers and the body can be sent with the actual content. If 8BITMIME is specified by the server, the content can be sent as a transparent 8-bit code; otherwise, only the 7-bit US-ASCII character set can

be used. The content is closed by a line with ".". The server replies with 250 Ok and a unique ID.

- QUIT tells the server that the transfer is finished. The server replies with 221 and terminates the transmission.

This is the sequence for an error-free e-mail transmission. However, various errors may occur, which can result in corresponding actions depending on the state of the transmission.

6.3.3 MIME



arrangement

6.3.3 MIME

6.3.3.1 Media Types

6.3.3.2 Content Transfer Encoding

The SMTP definition in RFC 822 assumes that only characters from the 7-bit US-ASCII code are used in e-mails. For example, umlauts cannot be displayed. The SMTP definition is extended with the **MIME (Multipurpose Internet Mail Extensions)** definitions (RFC 2045 to RFC 2049). This allows:

- that text messages can use codes other than US-ASCII,
- that different formats can be used for non-textual messages (pictures, audio, ...)
- that messages consist of several parts and,
- that characters other than those defined in the US-ASCII code can be used for headers.

The extensions are defined so that e-mails can still be processed according to the old SMTP standard.

The MIME standard defines a number of new header fields:

- **MIME version:** Mail programs can recognize whether the e-mail is composed according to the MIME standard. The default value is 1.0.
- **Content type:** It is used to describe the media type and subtype of the data in the e-mail. Information on how to display them may also be available.
- **Content transfer encoding:** This defines how the data of an e-mail was encoded or transformed.
- Other header fields, such as **content ID** or **content description**.



example

MIME version: 1.0

The e-mail is based on MIME version 1.0.

Content-type: text/plain; charset=US-ASCII

The content of the e-mail is a simple text without any special characters.

Content transfer encoding: 7BIT

A 7-bit code is used.

Content description: Mail message body

The content consists only of the body of the e-mail.

6.3.3.1 Media Types

The initial media types are described in RFC 2046. There are now some additional types registered at IANA (see [list at IANA](#)). The method for defining media types and subtypes is not only used by e-mail programs, but also by WWW browsers.

The original (and most important) types are:

- **Text:** Textual information. The following subtypes are used: *text/plain* for simple text, *text/enriched* or *text/richtext* for text with formatted characters (bold, italic, italic, etc.) or *text/html* for HTML-formatted e-mails.
- **Image:** Images, e.g. *image/tiff* or *image/jpeg* or *image/gif*.
- **Audio:** Audio data, e.g. *audio/x-pn-realaudio* for real audio files or *audio/basic* for audio files with the extension .au or .snd.
- **Video:** Video files, e.g. *video/mpeg* for mpeg files or *video/x-msvideo* for AVI files.
- **Application:** Other application programs, such as *application/octet-stream* for executable files or program code files, or *application/msword* for Word files.

New media types are included and defined in RFC 7303: *text/xml*, *application/xml*, *text/xml-external-parsed entity*, and *application/xml-dtd*.

In addition, there are media types that describe the **structure** of an e-mail:

- **Multipart:** The data consist of different parts, which can have different formats. The following subtypes exist:
 - *multipart/mixed*: for parts that are available in different formats
 - *multipart/alternative*: with different display types for the same content
 - *multipart/parallel*: if parts are to be viewed at the same time

- `multipart/digest`: if all parts are constructed according to RFC 822

In e-mails with multipart data it must be possible to distinguish between the parts. A **boundary** is introduced for this purpose. The specified value encloses a completed part of the e-mail.

- **Message**: An encapsulated message. The following subtypes exist:
 - `message/rfc822`: RFC 822 compliant data
 - `message/partial`: Data is too large to be transmitted at once, so it is sent in parts
 - `message/external-body`: Reference to external data

6.3.3.2 Content Transfer Encoding

The `content-transfer-encoding` header field describes how the data is encoded or transformed. If characters are to be transmitted where the bit can be not equal to 0 (e.g. umlauts or program files), the corresponding characters must be converted into the 7-bit US-ASCII code. The following encodings or transformations can be used for this purpose:

- **quoted-printable**
- **Base64**

The **quoted-printable** method is used if most of the characters are encoded according to US-ASCII and only a few other characters occur. Texts in this format are therefore more or less clearly readable.



example

An example from a trace can illustrate its use:

``

From: "Michael Praetorius" <praetorius@fh-luebeck.de>

Organization: FH Luebeck

To: praetorius@fh-luebeck.de

Date: Thu, 28 Mar 2002 11:41:37 +0100

MIME version: 1.0

Subject: ?ISO-8859-1?Q?Der_K=F6nig_ist_tot!?

Reply-to: praetorius@fh-luebeck.de

Message-ID: <3CA30171.16931.7B594D@localhost>

Priority: normal

X-mailer: Pegasus Mail for Windows (v4.01)

Content-type: text/plain; charset=ISO-8859-1

Content-transfer-encoding: Quoted-printable

Content-description: Mail message body

Es lebe der K=F6nig!

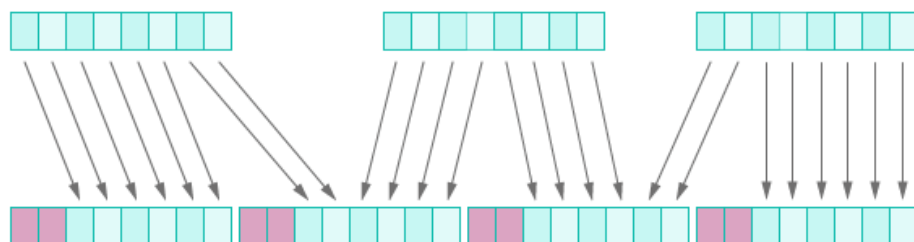
Both the header and the body contain an "ö" as non-US-ASCII character. The "ö" is recognized and replaced by the hex value F6.

The corresponding characters are encoded in the header according to the following format:

?ISO-8859-1?Q?Der_K=F6nig_ist_tot!?

First, the character table to be used is given; here it is ISO-8859-1. Then the coding method to be used is indicated: quoted printable or Base64; here it is quoted printable. Finally, the special characters are represented by the hexadecimal value from the character table, here F6, i.e. the 246th value from the ISO-8859-1 character table. The corresponding coded text is (with some effort) still readable.

The **Base64** method uses only 6 bits of the 7 bits allowed in the US-ASCII code. If we have four bytes available, we can use 24 bits (4 times 6 bits). From the original data, 3 bytes with 8 bits each are copied to the 4 bytes with 6 bits each. This gives us a completely illegible text, which is about 33% larger than the original text.



example

The following example from a trace can illustrate its use: The program ADDREG.EXE is transmitted as an attachment in a multipart e-mail. The text that separates the parts is:

Message-Boundary-22131

The example shows only a part of the mail:

MIME version: 1.0

Content-type: Multipart/Mixed; boundary=Message-Boundary-22131

--Message-Boundary-22131

Content-type: Application/Octet-stream; name="ADDREG.EXE"; type=PCEXE

Content-disposition: attachment; filename="ADDREG.EXE" Content-transfer-encoding: BASE64

TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAIAAAAAAAAAAAAAAAAAAAAAAAAAA

AAAAAAAAAgAAAAA4fug4AtAnNlbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW



RE9TIG1vZGUuDQ0KJAAAAAAAAABQRQAATAEGALQtPzkAAAAAAAAAAOACgELAQI8ACYAAA



--Message-Boundary-22131--

The bytes generated by the Base64 encoding are translated according to a special table – **Base64-Alphabet** – into printable characters; the characters that are of special importance for SMTP (such as “”, CR or LF) do not occur here.

6.3.4 Security

The e-mail protocols were designed without consideration of security aspects, so e-mails are transmitted unencrypted by default. The transmission of login and password is also unencrypted by default.

However, it is possible to combine the e-mail protocols with an SSL/TLS encryption, which is usually done today. However, this achieves only a partial protection of e-mails. If e-mails are transferred from the client to the mail server, they are encrypted (SMTP + SSL/TLS). On the mail server, however, the mails are unencrypted. When transferring between mail servers you as a user must trust that a further combination of SMTP with SSL/TLS will take place to secure the transmission. As the discussion in connection with the initiative [“E-mail made in Germany”](#)  showed, this was not the case in the past between the important providers (by the way, the legally binding [De-Mail](#)  is something different). On the mail server of the receiver, however, the mails are unencrypted. The retrieval of e-mails with POP or IMAP is now almost always combined with SSL/TLS.

If you want to perform an end-to-end encryption, you can use the standards "Pretty Good Privacy"  (offered for private users) or "S/MIME"  (within a company).

6.3.5 Mail Retrieval

As mentioned above, the reception of e-mails is more complicated than the sending of e-mails. We have already seen that three different methods are used to retrieve mail. All three procedures require that a mailbox is available at an ISP all the time.

While the mails are downloaded for offline reading with POP3, they remain in the mailbox at the ISP and are administered there if IMAP4 is used. This is also the case with web mail; here HTTP is used for the transmission. So you only need a browser and no special client.

IMAP4 and web mail now have the main advantage that e-mail management on the server avoids inconsistencies when accessing it via multiple clients. This is important since many users want to retrieve e-mails on the road via a smartphone or a notebook.

6.4 Domain Name System



arrangement

6.4 Domain Name System

6.4.1 DNS Name Space

6.4.2 Responsibilities

6.4.3 Database

6.4.4 Transfer Protocol

6.4.5 DNS Message Format

The Internet works on the basis of IP addresses, but as a human being these IP addresses are difficult to remember. Therefore, it is more convenient to use symbolic names and to automatically translate them into IP addresses.

The **DNS (Domain Name System)** (RFC 1034, RFC 1035) is a globally distributed database that allows host names to be mapped to associated IP addresses and vice versa. If, for example, the WWW server of FH Lübeck is to be addressed (www.fh-luebeck.de), the corresponding IP address must first be found (here: 193.175.120.224), since IP can

only process IP addresses and not host names. In addition, DNS is also required for e-mail.

Because it is impossible to store all current mappings on each host, a system has been developed that supports different mapping areas (zones). Zones are assigned to a company, a university (e.g. "fh-luebeck.de") or an organization, which maintains the current assignment tables independent from other organizations. Each zone must operate a DNS server.

If, for example, a host from Australia wants to address the WWW server of FH Lübeck, a request from Australia must first be sent to the DNS server at FH Lübeck responsible for the zone "fh-luebeck.de": "I want the IP address of the host www.fh-luebeck.de". The IP address is then sent to Australia. Only then can a connection from Australia to the WWW server be established.

The inverse case (**inverse DNS request**: "I want the hostname for the IP address 193.175.120.224") is not used as frequently and does not necessarily have to provide a result. The Internet applications can still work, since they know the IP address. For example, traceroute uses inverse queries to get the hostnames for the IP addresses of the routers.

When configuring a host, the DNS server for the zone that is to be the first contact for DNS requests must be specified. This information is usually configured automatically by DHCP.



annotation

Before the introduction of the DNS, a single file called hosts.txt was maintained centrally by the Network Information Center at the Stanford Research Institute and distributed periodically by FTP (see [Wikipedia entry on hosts](#)). As a remainder of it, you can still find such files in many operating systems, e.g. unter /etc/hosts in Linux.

6.4.1 DNS Name Space

The DNS name space is hierarchically structured which can be seen in the host names. Host names consist of multiple domains and the actual name of a host. The host name www.informatik.fh-luebeck.de consists of the **top-level domain (TLD)** "de" for Germany, the **sub-domain** "fh-luebeck" for FH Lübeck, the sub-domain "informatik" for the computer science department of FH Lübeck, and the **host name** "www" for a host named "www". A WWW service is probably offered on this host – the name "www" is an indication of this purpose, but this is not certain.

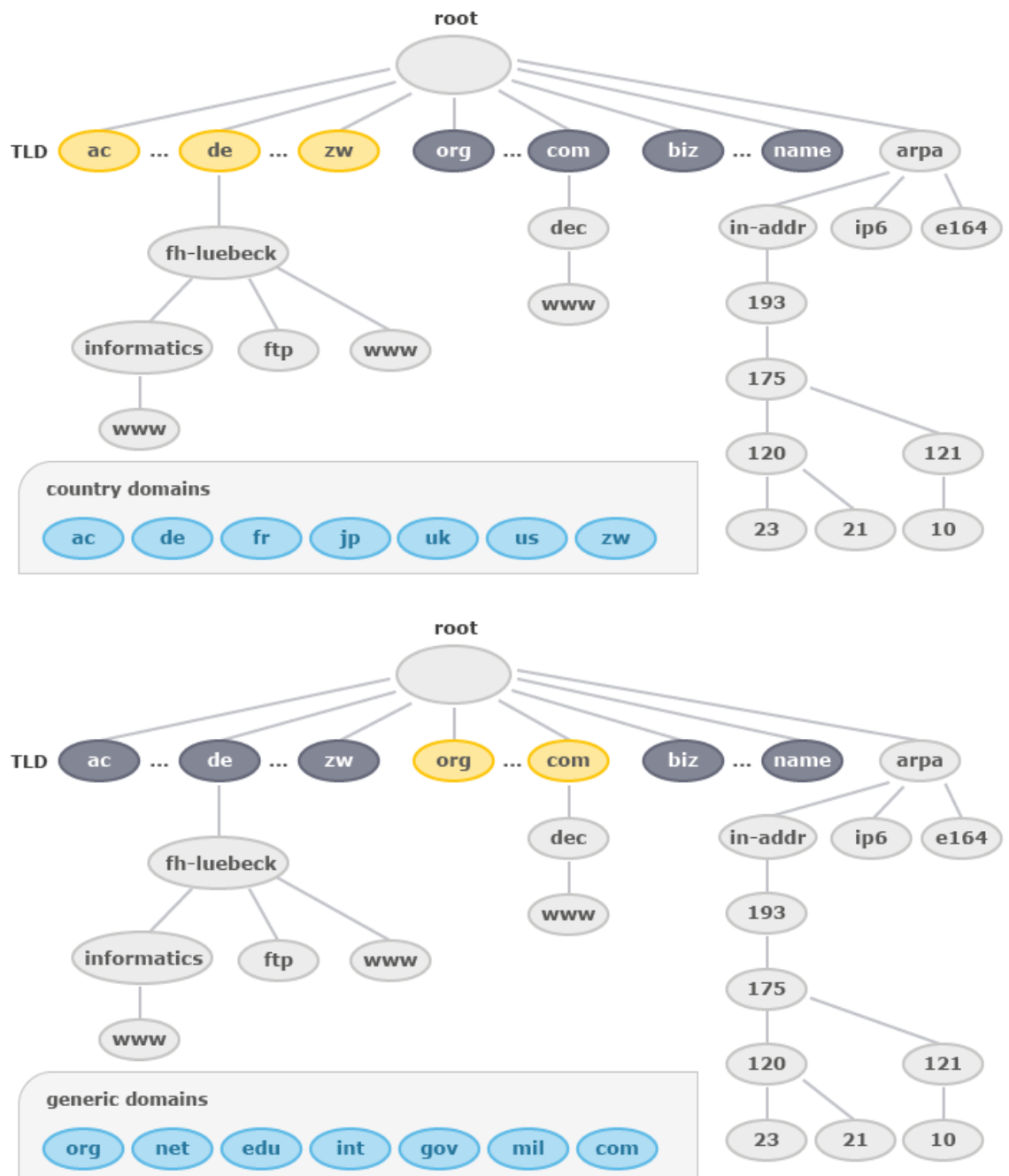
For an overview of the hierarchy, see the following figure:

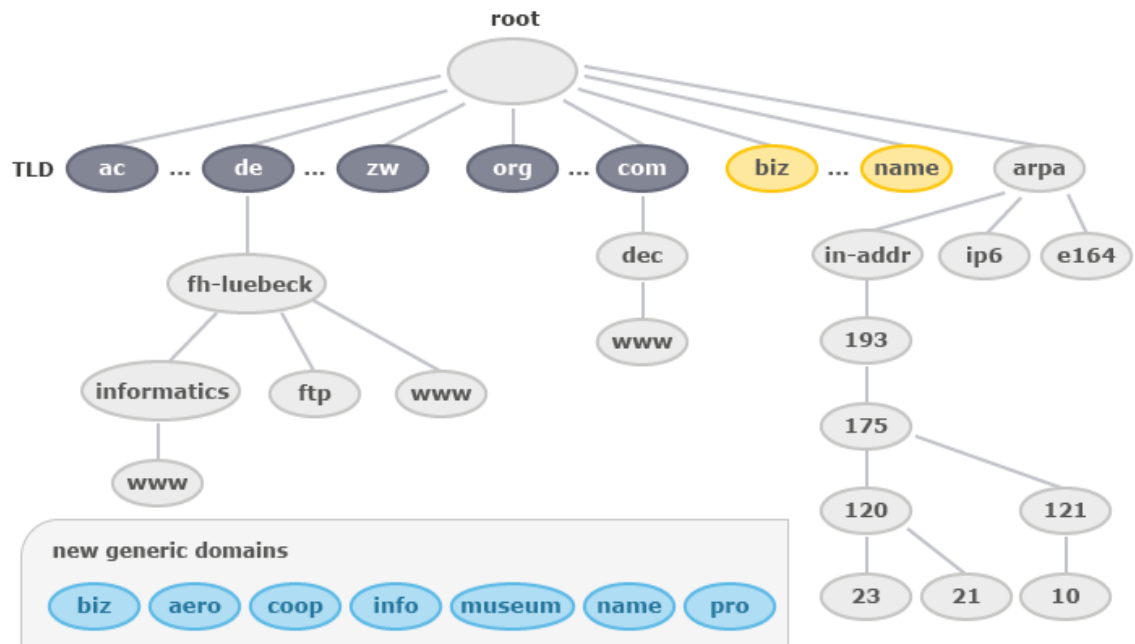


In the online version an rollover element is shown here.

DNS hierarchy

Begin printversion





End printversion

Among the top-level domains (see [complete list at IANA](#)) there are

- **Country domains** from “ac” for Ascension to “zw” for Zimbabwe. Some commonly used country domains are “de” for Germany, “jp” for Japan, “fr” for France, “uk” for United Kingdom and “us” for USA.
- **Generic domains:** The following seven domains were the first to be used in the Internet:
 - “org” for organizations
 - “net” for networks
 - “edu” for schools and colleges (educational)
 - “int” for international organizations
 - “gov” for government (USA only)
 - “mil” for military (USA only)
 - “com” for companies (commercial)
- Over time, many new generic domains have been introduced; some examples are:
 - “aero” for the aviation industry
 - “biz” for companies
 - “coop” for cooperative organizations
 - “info” without restriction
 - “museum” for museums
 - “name” for private individuals
 - “pro” for lawyers, tax consultants, doctors

- “berlin,” “hamburg” for the respective cities
- **Infrastructure domains**
 - “arpa” (address routing parameter area) is only used for infrastructure tasks.
 - “in-addr.arpa” is used for inverse queries (IPv4). The host names for given IPv4 addresses are stored in these domains. These assignments do not have to exist.
 - “ip6.arpa” is used for inverse queries (IPv6)
 - “E164.arpa” is for e.164 telephone numbers and services


Until 1 March 2004, no umlauts or other country-specific characters were allowed in DNS names. Today special characters are allowed in DNS names depending on the country. In Germany, in addition to the ASCII characters 92 special characters are allowed including all umlauts. These names, known as **IDN (Internationalized Domain Names)**, can be used, for example, as web or e-mail addresses. There are IDNs in more than 350 languages. This includes Korean, Greek, Russian, and Chinese domain names for example. Because DNS servers can still only understand ASCII characters, the country-specific special characters must be converted to ASCII characters before a DNS query can be made.

6.4.2 Responsibilities

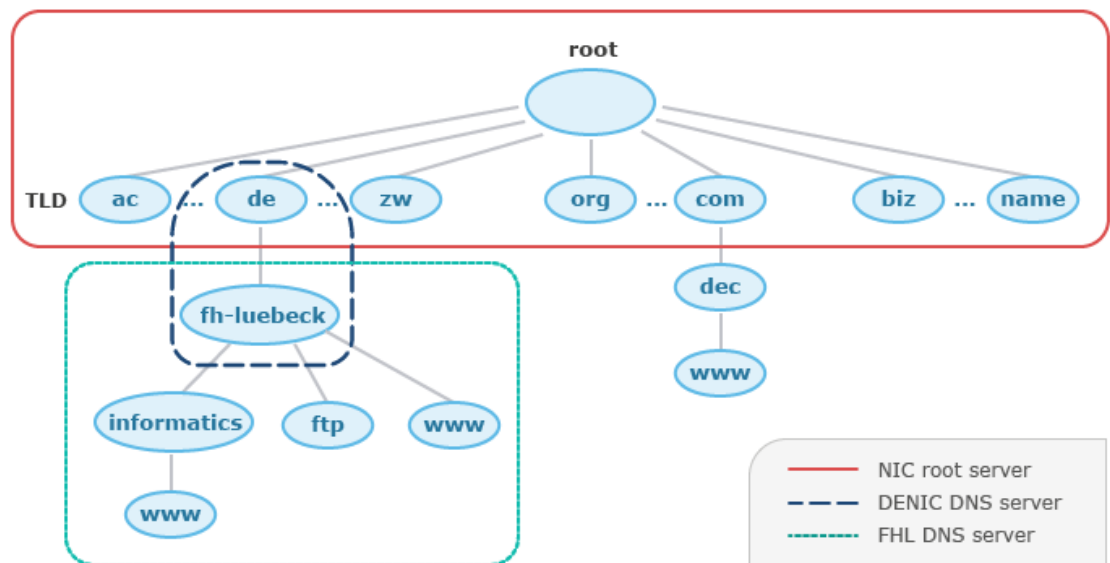
The responsibility for each zone belongs to an organization (universities, companies) that operate special DNS servers for this purpose. For availability reasons, each organization has a primary and a secondary DNS server. If a server fails, the second server answers the requests. If this one also fails, the hosts of the organization can no longer be reached via the host name (only via IP addresses) since the mapping tables of all hosts of the organization worldwide are stored on the DNS servers of the organization only once.



annotation

There are public DNS servers, such as the [Google](#)  servers with IPv4 addresses 8.8.8.8 and 8.8.4.4, which respond to queries about all possible zones. Their responses are, however, “non-authoritative,” so the assignment at the primary DNS server for the zone might be different.

In the DNS tables, the responsible domain is defined with a SOA (**Start of Authority**) entry. Thus the DNS server of FH Lüneburg is responsible for the domains “fh-luebeck.de” and the underlying domains – it does not know other domains.



Areas of responsibility of different DNS servers

If a DNS server cannot answer a request, it must ask another DNS server. But it cannot know all servers in the world – but it does not need to either. It only needs to know the IP addresses of the **DNS root servers** – currently these are 13 (logical) root servers – and send the request to one of them.

Each of the root servers knows the IP addresses of the DNS servers responsible for the various top-level domains (TLD). Thus the server `dns.denic.de` is responsible for “de”. The root server then sends the request with “de” to this server. The DENIC server knows all German DNS servers – e.g. the server of FH Lübeck, which is responsible for the zone “fh-luebeck.de” – and sends the request further. The DNS server of FH Lübeck is responsible for “fh-luebeck.de”. It finds the desired IP address in its assignment tables and returns its response to the requesting host.



In the online version an video is shown here.

Link to video : <http://www.youtube.com/embed/2epSZCKD57w>

DNS name resolution



webservice

Some [information](#) and [statistics](#) on the DNS for the .de domain and beyond are available at DENIC.

6.4.3 Database

A DNS server uses multiple files in which its database is defined. The entries in the files are called **resource records (RR)**.

An excerpt from the database for regular and inverse requests to the DNS server of FH Lünebeck with some important RRs is shown in the two rollover interactions below.



In the online version an rollover element is shown here.

Regular requests

Begin printversion

Name	Class	Type	
fh-luebeck.de.	IN	SOA	dns.fh-luebeck.de. herrmann.fh-luebeck.de. (200202072 ; Serial 21600 ; Refresh 1800 ; Retry 604800 ; Expire 172800) ; Minimum
	IN	NS	dns.fh-luebeck.de.
	IN	NS	deneb.dfn.de.
	IN	NS	ws-karl.win-ip.dfn.de.
fh-luebeck.de.	IN	A	193.175.120.17
deneb.dfn.de.	IN	A	192.76.176.9
ws-karl.win-ip.dfn.de.	IN	A	193.174.75.154
fh-luebeck.de.	IN	MX	10 noah.fh-luebeck.de.
	IN	MX	20 mx.fh-luebeck.de.
www.fh-luebeck.de	IN	A	193.175.120.99
localhost	IN	A	127.0.0.1

NS (Name Server)

NS defines the DNS servers responsible for the zone. Each zone requires at least two DNS servers.

field	description
SOA (Start of Authority)	SOA specifies the begin of a zone. The primary DNS server as well as the e-mail address of the person responsible for the server is provided (@ is replaced by .).
NS (Name Server)	NS defines the DNS servers responsible for the zone. Each zone requires at least two DNS servers.
A (Address)	Mapping of a DNS name to an IPv4 address. There should be an entry for each name here. Otherwise, the host is not reachable via its name.
MX (Mail Exchange)	Defines a mail server which can receive mail for a domain. Several MX records can be present for a domain. At first those mail servers are requested which have the lowest "preference" value (see column four).

End printversion



In the online version an rollover element is shown here.

Inverse requests

Begin printversion

Name	Class	Type	
120.175.193.in-addr.arpa.	IN	SOA	dns.fh-luebeck.de.
			herrmann.fh-luebeck.de.
			(200202050 ; Serial
			21600 ; Refresh
			3600 ; Retry
			604800 ; Expire
			172800) ; Minimum
99.120.175.193.in-addr.arpa.	IN	PTR	adam.fh-luebeck.de.
	IN	PTR	stud.fh-luebeck.de.
	IN	PTR	mail.fh-luebeck.de.
23.120.175.193.in-addr.arpa.	IN	PTR	eva.fh-luebeck.de.
	IN	PTR	ftp.fh-luebeck.de.
	IN	PTR	www.fh-luebeck.de.
1.0.0.127.in-addr.arpa.	IN	PTR	localhost.

field	description
SOA (Start of Authority)	SOA specifies the begin of a zone. The primary DNS server as well as the e-mail address of the person responsible for the server is provided (@ is replaced by .).
PTR (Domain Name Pointer)	Is used for inverse queries ("provide the name for a given IP address").

End printversion

The first column contains the name to be searched for, which is transmitted in a request. The second column indicates the class of the name. Here only IN (for Internet) is in use. The third column shows the type of the RR. The most important types are explained below. The fourth column contains the responses that are sent back to a request. They have different formats depending on the type of RR.

SOA (Start of Authority) SOA defines the beginning of a zone. The primary DNS server and the e-mail address of the person responsible for the server (. is replaced by @) are specified. **Serial** specifies the date of the last change in the format YYYYMMDDNN, where NN is a sequential number. **Refresh** (in seconds) indicates how often primary and secondary servers should align. **Retry** (in seconds) is used by the secondary server if a zone has been transmitted incorrectly. **Expire** (in seconds) indicates the upper limit for

using the data on the secondary server. **Minimum** specifies the lower value for the time that an entry is to be kept in the cache (time to live, **TTL**) if the TTL value is not defined in the individual RRs.

NS (Name Server) NS defines the servers responsible for the zone. Each zone requires at least two servers.

A (Address) Mapping of a DNS name to an IPv4 address. There should be an entry for each name; otherwise, the host cannot be addressed with the name.

AAAA Assignment of a DNS name to an IPv6 address (quad-A record).

CNAME (canonical name) Defines an alias or nickname for an official (or canonical) name.

HINFO (host info) Defines the hardware and operating system of a host.

PTR (domain name PoinTeR) Used for inverse requests (“give the name for an IP address”).

MX (Mail eXchange) Defines a host that can receive mail for a domain. There may be multiple MX records for a domain; the mail hosts that have the smallest value in the “preference” (fourth column) are addressed first.

TTL and minimum TTL and minimum affect only the distribution of changes and deletions in a zone – the distribution of newly recorded values is controlled via **refresh**.

The TTL value assigned to the individual records and the minimum TTL value assigned to the zone influence the network load and the response times to a considerable extent. Large values reduce the network load and shorten the response times. Small values lead to a faster exchange of database changes.

Typical values for TTL range from 0.5 to 7 days.

Root server The DNS server needs a list of all root servers (named.root) to specify its own zone. This can be found at internic.net [↗](#) (see also [IANA page](#) [↗](#)).

The 13 root servers were formerly individual physical servers. However, for a long time these servers are logical servers. In the background there are many physical servers which use anycast. You can find more information about the root servers and their locations on the root-servers.org [↗](#) page.

Caching To reduce the network load, all DNS servers have a cache. For each a request, an attempt is first made to find the response in the cache before the request is submitted to the next server. There are DNS cache-only servers that do not have their own database but only work with a cache.

6.4.4 Transfer Protocol

DNS uses the server port number 53. Its messages can be transmitted using UDP or TCP. UDP is usually used for regular, short DNS requests to generate less load and to achieve faster response times. TCP is used for safety reasons for transmission of zones between two DNS servers.

UDP messages cannot exceed 512 bytes (without header); otherwise, they are truncated. UDP messages can be lost. Therefore, a repetition mechanism must be available, but the repetitions should not be too numerous to avoid overloading the network. A waiting time of 2 to 5 seconds is recommended. Repeated requests should be sent to different DNS servers – not always to the same server, which may be unavailable.

6.4.5 DNS Message Format

According to RFC 1035 a DNS message consists of five parts:

Header	
Question	question for the name server
Answer	RRs answering the question
Authority	RRs pointing towards an authority
Additional	RRs with additional information



DNS message

Header The header is always present and has the following fields:

0	8	16	24	31
ID		Flags		
Number of questions		Number of RRs		
Number of name server RRs		Number of additional RRs		

**Header**

The ID identifies the message and is sent back to the requester by the server so that the request and response can be matched.



In the online version an rollover element is shown here.

Flag

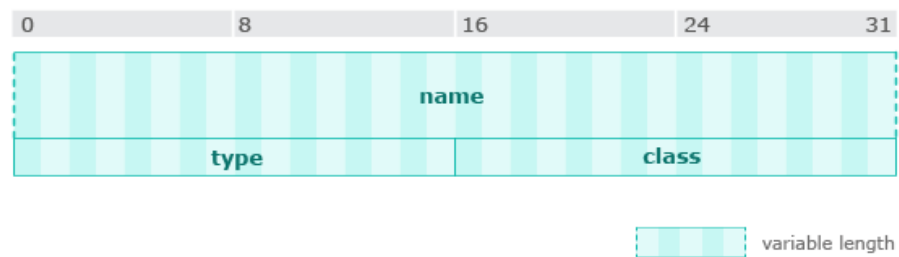
Begin printversion

0	8	15
QR	Opcode	AA TC RD RA Zero RCODE

field	description
QR (Query/Response)	0 = query, 1 = response
Opcode (4 Bit)	0 = standard query, 1 = inverse query, 2 = server status query
AA (Authoritative Answer)	1 = Server is responsible for the domain in the query
TC (Truncation)	The message was too long (more than 512 bytes in case of UDP) and was shortened
RD (Recursion Desired)	If 1, a recursive query is started. The requested name server forwards the request and expects an answer from another name server. If 0, an iterative query is started. The requested name server sends a list of other name servers back to the requester.
RA (Recursion Available)	The answer shows whether recursive requests are allowed on this name server.
Zero (3 Bit)	Is always 0
RCODE (Return Code, 4 Bit)	If return code = 0, no error occurred.

End printversion

Request As a rule, only one request is made. The most important field in a request is the name, which is variable in its length. It consists of a series of labels, each label consisting of an 8-bit length (permitted values 0 to 63) and the corresponding number of bytes (ASCII characters). The name is terminated by a length byte with value 0. The total length of a name cannot exceed 255 bytes.



Format of a request

The example shows how “www.fh-luebeck.de” appears in the request:



Format of the request to www.fh-luebeck.de

The name is followed by a 16-bit field, which indicates the type of the request (A record, type = 1 is most frequently searched). The type values for common RRs (see also the [Wikipedia list](#) [↗](#)) are:



In the online version an rollover element is shown here.

Type values for common RRs

Begin printversion

Type	Resource Record	field	description
1	A	A - Address	Mapping of a DNS name to an IPv4 address. An entry has to be available for each name. Otherwise, the host cannot be reached via its name.
28	AAAA	AAAA - Address	Mapping of a DNS name to an IPv6 address. The four As indicate that the IPv6 address length is four times the length of an IPv4 address.
2	NS	NS - Name Server	Defines the DNS servers responsible for the zone. Each zone requires at least two DNS servers.
5	CNAME	CNAME - Canonical Name	Defines an alias or nick name for an official (or canonical) name.
12	PTR	PTR - Domain Name Pointer	Are used for inverse queries ("give the name for an IP address").
13	HINFO	HINFO - Host Info	Defines hardware and operation system of a host.
15	MX	MX - Mail Exchange	Defines a server which can receive mails for the domain.

A - Address: Assignment of a DNS name to an IP address. There should be an entry for each name; otherwise, the host cannot be addressed with the name.

AAAA - Address: Mapping of a DNS name to an IPv6 address. The four As indicate that the IPv6 address length is four times the length of an IPv4 address.

NS - Name server: NS defines the server responsible for the zone. Each zone requires at least two DNS servers.

CNAME - Canonical name: Defines an alias or nickname for an official (or canonical) name.

PTR - Domain name pointer: Used for inverse requests ("give the name for an IP address").

HINFO - Host info: Defines the hardware and operating system of a host.

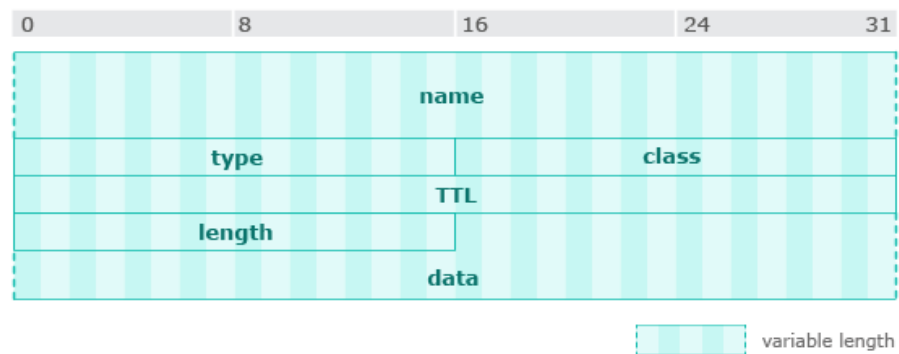
MX - Mail exchange: Defines a host that can receive mail for a domain.

End printversion

The last 16-bit field of the request contains the class: 1 means Internet; other values are unused.

Response All parts of the answers have the same format. They contain a variable number of RRs. The numbers are given in the header.

The responses begin with a variable-length name, the 16-bit long type, and the 16-bit class. The format of these values is identical to the format described in the question part.



Format of the response

This is followed by the 32-bit TTL (time to live) value, which indicates how long the RR may be kept in the cache before it has to be deleted. This is followed by a 16-bit length, which indicates the length of the subsequent data. If an A record (type = 1) has been requested, the data contains the 32-bit-long IP address.

The following examples show typical DNS requests:



example

IP address

What is the IP address of the host `www.fh-luebeck.de`?

The following trace gives us the IP address of `www.fh-luebeck.de`:

```
<span style="font-family: Courier New"> Domain name system (query)
```

```
Transaction ID: 0x0349
```

```
Flags: 0x0100 (Standard query)
```

```
0... .. = Query
```

```
.000 0... .. = Standard query
```

```
... ..0. .... = Message is not truncated
```

```
... ..1 .... = Do query recursively
```

```
... ..0 .... = Non-authenticated data is unacceptable
```

```
Questions: 1
```

```
Answer RRs: 0
```

```
Authority RRs: 0
```

```
Additional RRs: 0
```

```
Queries
```

```
www.fh-luebeck.de: type A, class inet
```

```
0000 00 b0 64 95 54 13 00 50 da 6d 1b 60 88 64 11 00 ..d.T..P.m.`d..
```

```

0010  03 7b 00 41 00 21 45 00 00 3f 07 50 00 00 80 11  .{.A.!E..?.P...
0020  fd eb 3e 9b bd 16 c1 af 78 11 04 d3 00 35 00 2b  ..>.....x....5.+
0030  ca ae 03 49 01 00 00 01 00 00 00 00 00 00 03 77  ...I.....w
0040
77 77 0a 66 68 2d 6c 75 65 62 65 63 6b 02 64 65
  www.fh-luebeck.de
0050  00 00 01 00 01  ....

```

The DNS message is marked in **red**.

 Domain name system (response)

transaction ID: 0x0349

Flags: 0x8580 (Standard query response, No error)

1... .. = Response

.000 0... .. = Standard query

... .1.. .. = Server is an authority for domain

... .0. = Message is not truncated

... ..1 ... = Do query recursively

... .. 1... .. = Server can do recursive queries

... .. .0. = Answer/authority portion was not authenticated

... .. 0000 = No error

Questions: 1

Answer RRs: 1

Authority RRs: 0

Additional RRs: 0

Queries

www.fh-luebeck.de: type A, class inet

Answers

www.fh-luebeck.de: type A, class inet, addr 193.175.120.23

Name: www.fh-luebeck.de

Type: Host address

Class: inet

Time to live: 2 days

Data length: 4

Addr: 193.175.120.23

```

0000  00 50 da 6d 1b 60 00 b0 64 95 54 13 88 64 11 00  .P.m.`..d.T.d..
0010  03 7b 00 51 00 21 45 00 00 4f 60 2c 00 00 36 11  .{.Q.!E..O`.,.6.
0020  ee ff c1 af 78 11 3e 9b bd 16 00 35 04 d3 00 3b  ....x.>....5...;
0030  69 70 03 49 85 80 00 01 00 01 00 00 00 00 03 77  ip.I.....w

```



```

0040  77 77 0a 66 68 2d 6c 75 65 62 65 63 6b 02 64 65  ww.fh-luebeck.de
0050  00 00 01 00 01 c0 0c 00 01 00 01 00 02 a3 00 00  .....
0060  <span style="color:green;">04 c1 af 78 17

```

The DNS message is **red**; the IP address is **green**.



example

Host name

Of course, we can also ask for the name of a host. So what is the name of host 193.175.12.23?

 Domain name system (query)

Transaction ID: 0x03c2

Flags: 0x0100 (Standard query)

0... .. = Query

.000 0... .. = Standard query

... ..0. = Message is not truncated

... ..1 ... = Do query recursively

... ..0 ... = Non-authenticated data is unacceptable

Questions: 1

Answer RRs: 0

Authority RRs: 0

Additional RRs: 0

Queries

23.120.175.193.in-addr.arpa: type PTR, class inet

Name: 23.120.175.193.in-addr.arpa

Type: Domain name pointer

Class: inet

```

0000  00 b0 64 95 54 13 00 50 da 6d 1b 60 88 64 11 00  ..d.T..P.m.`d..
0010  03 7b 00 4b 00 21 45 00 00 49 07 58 00 00 80 11  .{.K.!E..IX....
0020  fd d9 3e 9b bd 16 c1 af 78 11 04 d7 00 35 00 35  ..>.....x. 5.5
0030  13 51 03 c2 01 00 00 01 00 00 00 00 00 00 02 32  .Q.....2
0040  33 03 31 32 30 03 31 37 35 03 31 39 33 07 69 6e  3.120.175.193.in
0050  2d 61 64 64 72 04 61 72 70 61 00 00 0c 00 01  -addr.arpa.....

```

The DNS message is marked in **red**.

There are several names for the desired IP address.

 Domain name system (response)

Transaction ID: 0x03c2
 1... .. = Response
 .000 0... .. = Standard query
1.. .. = Server is an authority for domain
0. = Message is not truncated
1 = Do query recursively
 1... .. = Server can do recursive queries
0. = Answer/authority portion was not authenticated
 0000 = No error

Questions: 1

Answer RRs: 6

Authority RRs: 0

Additional RRs: 0 Queries

23.120.175.193.in-addr.arpa: type PTR, class inet

Name: 23.120.175.193.in-addr.arpa

Type: Domain name pointer

Class: inet

Answers

23.120.175.193.in-addr.arpa: type PTR, class inet, ptr eva.fh-luebeck.de

23.120.175.193.in-addr.arpa: type PTR, class inet, ptr eva1.fh-luebeck.de

23.120.175.193.in-addr.arpa: type PTR, class inet, ptr eva100.fh-luebeck.de

23.120.175.193.in-addr.arpa: type PTR, class inet, ptr ftp.fh-luebeck.de

23.120.175.193.in-addr.arpa: type PTR, class inet, ptr mx.fh-luebeck.de

23.120.175.193.in-addr.arpa: type PTR, class inet, ptr www.fh-luebeck.de



example

Mail server

Finally, let's look at a third trace. Which mail servers should be addressed for the e-mail domain fh-luebeck.de?

Domain name system (response)

transaction ID: 0x0229

Flags: 0x0100 (Standard query)

Questions: 1

Answer RRs: 0

Authority RRs: 0

Additional RRs: 0

Queries

fh-luebeck.de: type MX, class inet

In response, two hosts are specified with different **preference**.

 Domain name system (response)

Transaction ID: 0x0229

Flags: 0x8580 (Standard query response, No error)

Questions: 1

Answer RRs: 2

Authority RRs: 0

Additional RRs: 2 Queries

fh-luebeck.de: type MX, class inet

Answers

fh-luebeck.de: type MX, class inet, preference 10, mx noah.fh-luebeck.de

Name: fh-luebeck.de

Type: Mail exchange Class: inet

Time to live: 2 days Data length: 9

Preference: 10

Mail exchange: noah.fh-luebeck.de

fh-luebeck.de: type MX, class inet, preference 20, mx mx.fh-luebeck.de

6.5 Exercises - Application Layer



task

Tasks for Beginners

Task 1:

Use the command line tool [nslookup](#) to perform DNS queries. Under Linux you can use [dig](#) as an alternative.

- Find out which IP address the (primary) DNS server of FH Lübeck has.
- Record DNS requests with Wireshark and find out which Transport Layer protocol is used by DNS.

Task 2:

For years, it is common that commercial Internet sites are not realized just by using a single web server. Use the Firefox AddOn [IPvFox](#) or the Chrome AddOn [IPvFoo](#) to determine how many servers are used when accessing the website www.t-online.de.

Remark: Please take care that the use of other AddOns (e.g. an ad blocker) or caching do not influence the result.

Task 3:

Use webpagetest.org to examine the edx.org web page. Take a look at the different result tabs.

Tasks for Advanced Learners**Task 1:**

Examine the [HTTP/2 test page of Akamai](http://http2.test.akamai.com) using the developer tools (tab "Network") in the Google Chrome browser. Pay particular attention to the summarized temporal display.

Remark: In addition to the developer tools it is also possible to get information about the communication over the network by accessing "chrome://net-internals".

Task 2:

Use the [IIS test page](http://iis.test.fh-luebeck.de) (this organization is responsible for the administration of the .se domains) to check the configuration of the FH Lübeck DNS server. Please consider also the "Advanced Results" tab.

Task 3:

Change the DNS configuration on your computer so that you are using a Google DNS server. Take note of instructions provided by [Google](https://www.google.com/dns/).

Task 4:

The last task gives some ideas if you want to learn more about the server side of applications.

- The Apache Server is the most frequently used web server. The web page [apachefriends](http://httpd.apache.org/) provides installation packages for different operating systems. By using [Let's encrypt](https://letsencrypt.org/) you can also get familiar with SSL/TLS certificates.
- For experiments related to mail servers you can install a local [hMailServer](http://hmailserver.org/).
- To try out the DNS you can run the most frequently used DNS software [BIND](http://isc.org/bind/). As an alternative it is also possible to use the DNS functionality of [pfSense](http://pfsense.org/) or [OPNsense](http://opnsense.org/) (see [comparison web page](http://www.comparisons.com/)).

6.6 Summary - Application Layer

In this chapter, WWW, e-mail and DNS were presented as selected examples of Internet applications. There are, of course, many other applications, such as file transfers (FTP), voice over IP (VoIP, protocols RTP/RTCP, SIP), or messenger services.