Computer Networking and IT Security (CNS)

INHN0012 – WiSe 2025/26

Lorenz Lehle, Stephan Günther

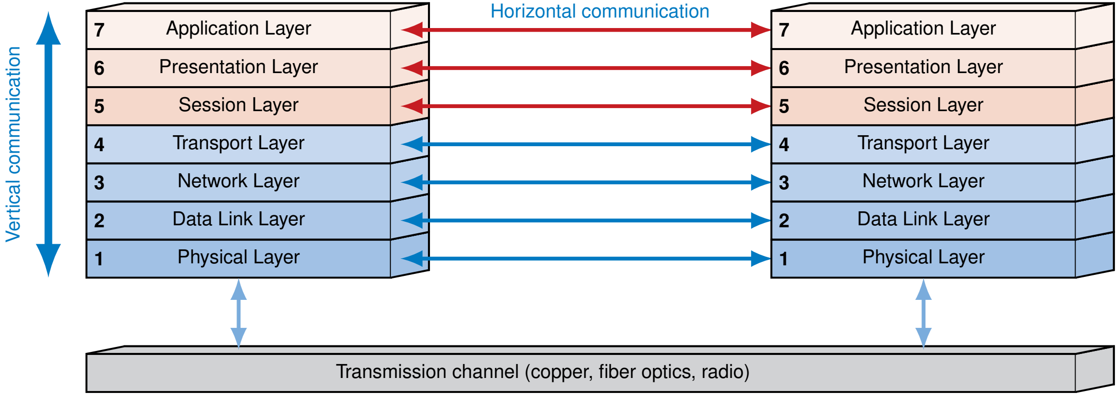
Chair of Distributed Systems and Security
School of Computation, Information and Technology
Technical University of Munich

Wednesday 21st January, 2026 13:35

Chapter 5: Session, Presentation, and Application Layer

Chapter 5: Session, Presentation, and Application Layer

Placement in the ISO/OSI model



- Session and presentation layer services are implemented in individual cases in the form of standardized protocols.
- In many cases, the functions of the session layer or the presentation layer are integrated into the respective application.

$\Rightarrow$ It is often difficult to differentiate

Placement in the ISO/OSI model

Model and reality

- The ITU[1] X.200 series standards describe services of the seven OSI layers as well as protocols for providing these services.
- The structuring done in these standards is useful.
- However, several OSI protocols have hardly any significance in practice.
- Often no strict separation between session, presentation, and application layer is possible.
- In the Internet Model (RFC 1122)[2], all these functions are assigned to the application layer.

In the following we will

- explain the tasks of the session and presentation layer,
- get to know, by way of examples, some protocols whose functions can be assigned to layers 5 – 7, and
- explain important protocols of the application layer.

[1] International Telecommunication Union

[2] The Internet model is an alternative layer model (cf. ISO/OSI model) in which layers 1 and 2 as well as 5-7 are combined in each case. This corresponds in many cases to a more practical classification, although the summary sometimes represents a strong simplification. However, the latter even applies to individual layers of the OSI model, e. g. within layer 2 media access control can be clearly distinguished from addressing, which is why we sometimes speak of MAC and LLC (Logical Link Control) sublayers.

Chapter 5: Session, Presentation, and Application Layer

Session Layer

According to the ITU-T X.200 standard[1], session layer services are either connection-oriented or connectionless:

Connection-oriented

- A connection is established between the communication partners.
- The connection is maintained over the duration of individual transfers (or connections) of the transport layer.
- As with TCP connections, a distinction can be made between the connection establishment, data transfer and connection termination phases.

Connection-less

- Data is essentially only passed from the transport layer to higher layers and vice versa.
- No connection is established and no state is held between the communication partners.

Note
A session layer connection (session) is not the same as a transport layer connection. For example, a session can contain several TCP connections in succession or add functions such as confirmations to UDP-based applications, which in the case of TCP are automatically provided by the transport layer.

[1] `https://www.itu.int/rec/T-REC-X.200-199407-I`

Services on the Session Layer

Session
A session describes the communication between at least two participants with defined beginning and end as well as resulting duration.

In order to enable dialog guidance for the layer taking a service (here the presentation layer), several transport layer connections may have to be used and controlled. This may include handling of dropped and resumed TCP connections.

In connection-oriented mode, various services are offered:

- establishment and termination of sessions,
- normal and expedited data transfer[1],
- token management for coordination of session members,
- synchronization and re-synchronization,
- error messages / handling and activity management, as well as
- keep-alive and resume of sessions after disconnects.

[1] Expedited data transfer may be used for urgent data, e. g. alerts or interrupts.

Real-world examples

Example 1: HTTP (Hypertext Transfer Protocol)[1] → more on that later

- HTTP in its original form is initially stateless.
- There is therefore no connection between different requests and responses.
- Cookies allow a session to persist across multiple requests and responses, interactions, and TCP connections.
- This makes it possible to assign requests that are separated in time or originate from different addresses to a specific client and thus to a specific user.
- HTTP is usually assigned to the application layer (layer 7), but also includes functions of the presentation and session layers (layers 6/5).

[1] HTTP is an application layer protocol.

Real-world examples

Example 2: TLS (Transport Layer Security)[1]

- TLS is a protocol for authentication and encrypted transmission of data over a connection-oriented transport service.
- It is the basis, for example, for HTTPS.
- Among others, it offers in particular
  - authentication[2] (“Is my counterpart who I think he is?”),
  - data integrity (protection against data manipulation), and
  - encryption (confidentiality, i. e., protection against unauthorized reading of data).
- During connection establishment, the communication parameters of the session (in particular authentication and encryption procedures as well as certificates) are exchanged.
- Sessions remain active over multiple subsequent TCP connections.
- While the TLS functions for session management and resumption are assigned to the session layer, the encryption functions can be assigned to the presentation layer.

[1] Serves only as an example here, but will not be discussed in more detail in the course of the lecture. However, one should know what it is for.

[2] Not to be confused with authorization, i. e., access rights verification.

Chapter 5: Session, Presentation, and Application Layer

Presentation layer

The task of the presentation layer is to enable the communication partners to interpret the data in a common way, i. e., to transmit data in a common format.

The following tasks are assigned to the presentation layer:

- representation of data and its syntax
- data structures for transmission of data
- representation of actions on those data structures
- transformation of data

Note
The representation on layer 6 does not have to correspond to the representation on layer 7 (application layer). The presentation layer is responsible for the syntax of the payload, the semantic remains with the applications.[1]

- Applications should be able to communicate with each other independently of syntax.
- Application-specific syntax can be converted into a common form by the presentation layer and then transferred.

[1] Syntax is the representation of data according to certain rules (grammar). If data is assigned a meaning, one speaks of information (task of semantics).

Tasks of the Presentation Layer

Concrete functions can be assigned to the basic tasks of the presentation layer:

- Encoding of data
  - translation between different character sets and codewords according to standardized encoding procedures
  - compression of data before transmission (removal of unstructured redundancy)
  - encryption
- Structured representation of data
  - platform-independent, consistent representation
  - translation between different data formats
  - serialization of data before transmission

As with the session layer, the same applies here: protocols that can be assigned to the presentation layer often also fulfill functions of other layers.

Example: TLS

- The functionality to encrypt data can be assigned to the presentation layer.
- However, functionality such as connection establishment, negotiation of authentication and encryption parameters, authentication, and authorization are commonly assigned to the session layer.

Character sets and data encoding

Data may be represented in two different formats:

(1) Text characters or symbols in human readable form, e. g. letters, text representation of numbers, special characters . . .

- A character set is a set of textually representable characters and their assignment to a codepoint[1].
- How the codepoints of a given character set are represented in binary form, i. e., by a sequence of bits, is determined by encoding rules.
- There may also be different encoding rules for the same character set.

(2) Binary data, i. e., a sequence of bits, e. g. the binary representation of letters, numbers, and symbols, but also images, music, movies, etc. in digital form.

- A datum is a computer-processable “unit”, i. e., a short sequence of bits whose length is usually a multiple of 8 bit.[2]
- What a datum represents – a number, a character, part of a number, or even an image – depends on the context (see Chapter 1 “information and its meaning”).
- If it is known that the data is text, what encoding was used, and what character set it is, the binary data can be easily translated back into text characters.

Note: Binary data, e. g. an image, cannot be easily represented with a character set. For this purpose, there are special encoding rules that make it possible to encode binary data so that it can be represented with a character set.

[1] A codepoint is a unique “identifier” for at most one text character (not all codepoints need to be assigned).

[2] The smallest addressable unit in memory for today’s computers is an octet, i. e., a block of 8 bit, which is commonly called a byte. Other common sizes are 16 bit, 32 bit and 64 bit, which are often called word, double word, and quad word, respectively. Memory is addressed byte by byte, whereby it depends on the processor in which order the individual octets are loaded. Consequently, the byte order is essential.

Character sets and data encoding

- A character set consists of a set of characters where each character is associated with a codepoint.
- Examples for character sets are ASCII, ISO-8859-15 (ISO-8859-1 which contains €), and Unicode.
- A character set may consist of printable characters as well as control characters.
- ASCII defines 128 characters:
  - The characters with codepoints 0 – 31 and 127 are control characters.
  - Examples for such control characters are line breaks, tabulator, and protocol characters for data transfer.
  - Originally printers and terminals were controlled this way.
- ISO-8859-15 defines 256 codepoints:
  - The codepoints 0 – 127 correspond to the ASCII character set.
  - Further codepoints are linked with special characters of European languages as well as the Euro sign.
  - ISO-8859-15 was a widely used character set in the Western European language area before Unicode.

Character sets and data encoding

- A character set consists of a set of characters where each character is associated with a codepoint.
- Examples for character sets are ASCII, ISO-8859-15 (ISO-8859-1 which contains €), and Unicode.
- A character set may consist of printable characters as well as control characters.
- ASCII defines 128 characters:
  - The characters with codepoints 0 – 31 and 127 are control characters.
  - Examples for such control characters are line breaks, tabulator, and protocol characters for data transfer.
  - Originally printers and terminals were controlled this way.
- ISO-8859-15 defines 256 codepoints:
  - The codepoints 0 – 127 correspond to the ASCII character set.
  - Further codepoints are linked with special characters of European languages as well as the Euro sign.
  - ISO-8859-15 was a widely used character set in the Western European language area before Unicode.
- Unicode
  - Unicode in version 13.0 defines 143 859 different characters.
  - The total amount of possible codepoints is 1 114 112, structured in 17 levels with 2^{16} codepoints each.
  - Unicode aims to represent all writing cultures and character systems.
  - The codepoints 0 – 255 correspond to ISO-8859-1.
  - Unlike restricted character sets such as ASCII and ISO-8859-15, the Unicode character set is regularly updated and extended.
  - Several character encodings are defined which include codepoints for all Unicode characters used to map the codepoints to concrete binary representations, including UTF-32LE (little endian), UTF-32BE (big endian), UTF-16LE and UTF-16BE, and UTF-8.

Encoding

Characters (respectively their codepoints) must be encoded for transmission. We can differentiate between

- fixed-length codes where all characters are encoded by codewords of the same length, e. g. ASCII (7 bit) or UCS-2 (16 bit), and
- variable-length codes where characters are encoded by codewords of different length, e. g. UTF-8 (1 B to 4 B).[1]

The possible encoding methods depend on the respective character set:

- ASCII and ISO-8859-15 define the encoding together with the character set.
  - ASCII codewords are 7 bit long, where the highest-order bit of an octet is always `0`.
  - ISO-8859-15 uses 8 bit long code words.
- Unicode does not define an encoding but only a character set. Often UTF-8 is used to encode Unicode, which is compatible (with some limitations) with ASCII[2].

[1] Morse characters are another example: here, more frequently occurring characters are assigned to shorter code words.

[2] The codepoints 0 – 127 correspond to ASCII.

Unicode Transformation Format (UTF-8)

UTF-8 encodes the Unicode character set with 1 B to 4 B depending on the respective codepoint:

Unicode range	length	binary UTF-8 encoding	encodable bits
`U+0000 - U+007F`	1 B	`0xxxxxxx`	7
`U+0080 - U+07FF`	2 B	`110xxxxx 10xxxxxx`	11
`U+0800 - U+FFFF`	3 B	`1110xxxx 10xxxxxx 10xxxxxx`	16
`U+10000 - U+1FFFFF`	4 B	`11110xxx 10xxxxxx 10xxxxxx 10xxxxxx`	21

- The representation `U+xxxx` is merely a notation of code points for Unicode.
- The hexadecimal digits indicate the value of the encoded bits of a codeword.
- For codewords longer than 1 B, the number of leading `1` bits before the first `0` bit in the first octet indicates the length of the codeword.
- The two highest-order bits of all subsequent octets of a codeword are `10`.
- For code words consisting of only one octet, the highest-order bit is always `0` (cf. ASCII).

Properties:

- UTF-8 is backward compatible with ASCII: ASCII encoded text is valid UTF-8 and can be interpreted as Unicode without conversion.
- UTF-8 is prefix-free[1] and therefore self-synchronizing.

[1] No valid codeword is true prefix of another codewords.

Encoding

Example 1: Encoding of the Umlaut ä

- Unicode: codepoint 228, codeword in UTF-8: `U+00E4 = 11000011 10100100`
  (“LATIN SMALL LETTER A WITH DIAERESIS”)
- ISO-8859-1 and ISO8859-15: codepoint 228, codeword: `11100100`
  Although both character sets are a subset of Unicode, the Unicode character encoding results in different code words.
- ASCII: no encoding possible since letters with diaeresis are not part of the character set defined by ASCII.

Example 2: Encoding of the EUR symbol €

- Unicode: codepoint 8364, codeword in UTF-8: `U+20AC = 11100010 10000010 10101100`
- ISO-8859-15: codepoint 164, codeword `10100100`
- ISO-8859-1 and ASCII: no encoding possible

Encoding

Example 1: Encoding of the Umlaut ä

- Unicode: codepoint 228, codeword in UTF-8: `U+00E4 = 11000011 10100100`
  ("LATIN SMALL LETTER A WITH DIAERESIS")
- ISO-8859-1 and ISO8859-15: codepoint 228, codeword: `11100100`
  Although both character sets are a subset of Unicode, the Unicode character encoding results in different code words.
- ASCII: no encoding possible since letters with diaeresis are not part of the character set defined by ASCII.

Example 2: Encoding of the EUR symbol €

- Unicode: codepoint 8364, codeword in UTF-8: `U+20AC = 11100010 10000010 10101100`
- ISO-8859-15: codepoint 164, codeword `10100100`
- ISO-8859-1 and ASCII: no encoding possible

Characters not supported by the used character set can be encoded by using character entity references:

- For example, XML allows the encoding of arbitrary Unicode characters by specifying the code point, e. g. `&#228;`
- HTML allows encoding of more commonly used characters using named entities, e. g. `&auml;`
- LaTeX allows to encode different characters in a similar way, e. g. `\"a`[1]

However, the syntax depends on the protocol used (HTTP, SMTP → more on that later) or the application.

[1] If one mistypes `Verschl\"usselung` and types `Verschlu\"sselung`, the result in the slides is Verschlüsselung.

Data compression

With respect to data compression, we have to differentiate between two fundamentally different approaches:

1. Lossless compression
   - Compressed data can be recovered without loss, i. e. accurately and without loss of information.
   - Lossless compressed formats are, for instance, ZIP, PNG[1] (images), FLAC[2] (music), . . .

2. Lossy compression:
   - Compressed data generally cannot be reconstructed accurately.
   - There is a loss of information during compression.
   - However, lossy compression may allow for a higher compression ratio or variable compression rate.
   - Lossy compressed formats are, for instance, MP3, MPEG, JPEG, . . .

[1] Portable Network Graphics

[2] Free Lossless Audio Codec

Data compression

Huffman Code

Many protocols compress data before sending (source coding).

In many cases, file formats use the Huffman code as backend or in combination with other algorithms, e. g. ZIP / DEFLATE, JPEG.

Basic idea of Huffman encoding:

- Not all text characters occur with the same frequency, e. g. the letter "E" occurs in the German language with a frequency of 17.4 %, followed by "N" with 9.8 %.
- Instead of encoding characters with codewords of the same length (e. g. ASCII), more frequently occurring characters are assigned to shorter codewords.
- The mapping between characters and codewords remains unique and reversible, which is why it is a lossless compression method.

Data compression

Huffman code

Constructing a Huffman code

- Given the alphabet $\mathcal{A} = \{A,B,C,D,E,F,G,H\}$ and probabilities of occurrence $\Pr[X = z]$ for all characters $z \in \mathcal{A}$.
- We further require that individual occurrences of characters are independent from each other.[1]

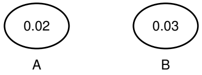
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30

[1] Otherwise, the statements about the optimality of the Huffman code in general would no longer apply.

Constructing a Huffman code

- Given the alphabet $\mathcal{A} = \{A,B,C,D,E,F,G,H\}$ and probabilities of occurrence $\Pr[X = z]$ for all characters $z \in \mathcal{A}$.
- We further require that individual occurrences of characters are independent from each other.

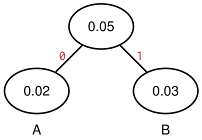
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Constructing a Huffman code

- Given the alphabet $\mathcal{A} = \{A,B,C,D,E,F,G,H\}$ and probabilities of occurrence $\Pr[X = z]$ for all characters $z \in \mathcal{A}$.
- We further require that individual occurrences of characters are independent from each other.

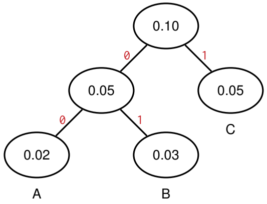
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Constructing a Huffman code

- Given the alphabet $\mathcal{A} = \{A,B,C,D,E,F,G,H\}$ and probabilities of occurrence $\Pr[X = z]$ for all characters $z \in \mathcal{A}$.
- We further require that individual occurrences of characters are independent from each other.

z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



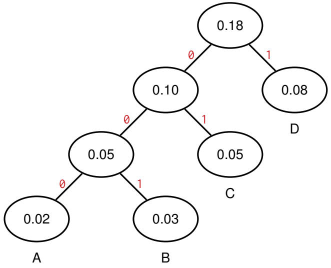
Data compression

Huffman code

Constructing a Huffman code

- Given the alphabet $\mathcal{A} = \{A,B,C,D,E,F,G,H\}$ and probabilities of occurrence $\Pr[X = z]$ for all characters $z \in \mathcal{A}$.
- We further require that individual occurrences of characters are independent from each other.

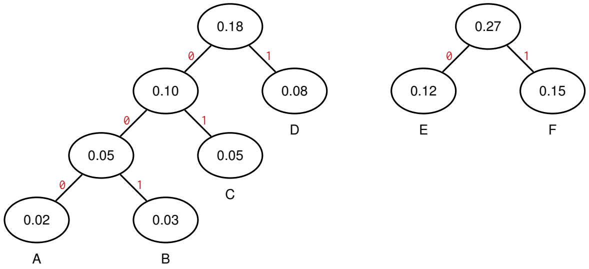
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Constructing a Huffman code

- Given the alphabet $\mathcal{A} = \{A,B,C,D,E,F,G,H\}$ and probabilities of occurrence $\Pr[X = z]$ for all characters $z \in \mathcal{A}$.
- We further require that individual occurrences of characters are independent from each other.

z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



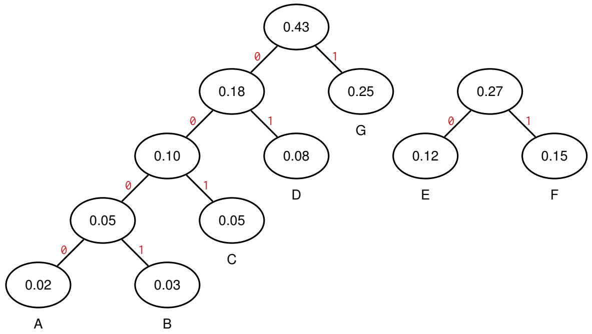
Data compression

Huffman code

Constructing a Huffman code

- Given the alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ and probabilities of occurrence $\Pr[X = z]$ for all characters $z \in \mathcal{A}$.
- We further require that individual occurrences of characters are independent from each other.

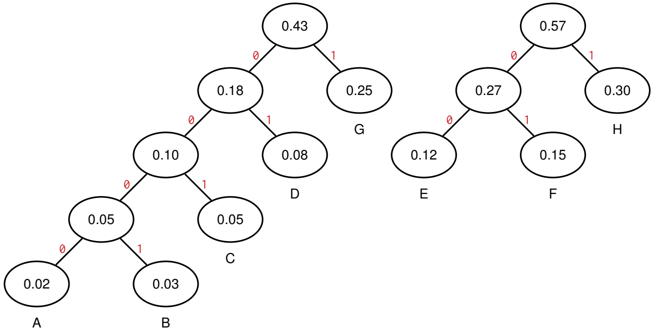
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Constructing a Huffman code

- Given the alphabet $\mathcal{A} = \{A,B,C,D,E,F,G,H\}$ and probabilities of occurrence $\Pr[X = z]$ for all characters $z \in \mathcal{A}$.
- We further require that individual occurrences of characters are independent from each other.

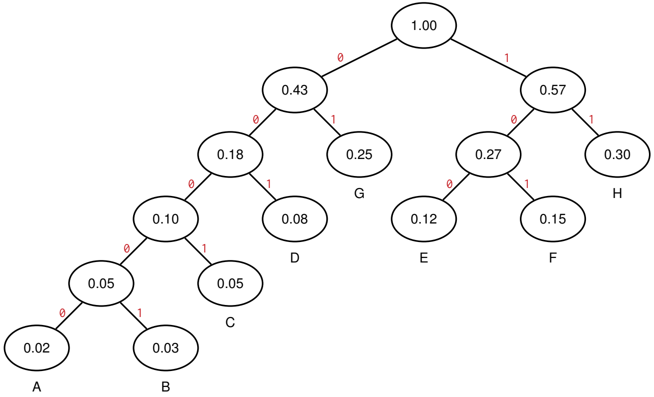
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Constructing a Huffman code

- Given the alphabet $\mathcal{A} = \{A,B,C,D,E,F,G,H\}$ and probabilities of occurrence $\Pr[X = z]$ for all characters $z \in \mathcal{A}$.
- We further require that individual occurrences of characters are independent from each other.

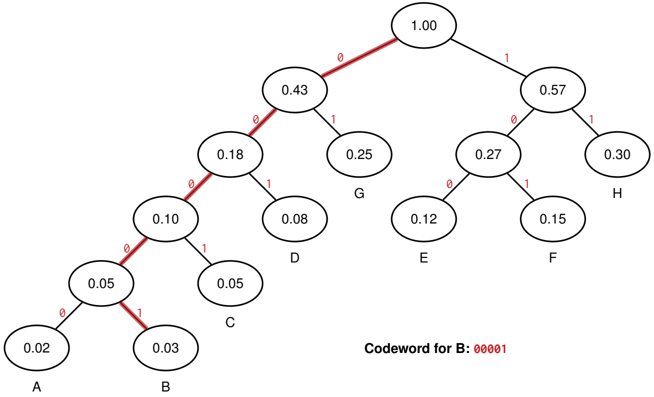
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Constructing a Huffman code

- Given the alphabet $\mathcal{A} = \{A,B,C,D,E,F,G,H\}$ and probabilities of occurrence $\Pr[X = z]$ for all characters $z \in \mathcal{A}$.
- We further require that individual occurrences of characters are independent from each other.

z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Codeword for B: `00001`

Data compression

Huffman-Code

Average length of codewords

z	$\Pr[X = z]$	Huffman code	length $l_H(z)$	uniform code
A	0.02	`00000`	5	`000`
B	0.03	`00001`	5	`001`
C	0.05	`0001`	4	`010`
D	0.08	`001`	3	`011`
E	0.12	`100`	3	`100`
F	0.15	`101`	3	`101`
G	0.25	`01`	2	`110`
H	0.30	`11`	2	`111`

- Uniform code: $\mathrm{E}[l_U(z)] = 3.0$, since all codewords have the same length
- Huffman code: $\mathrm{E}[l_H(z)] = \sum_{z \in \mathcal{A}} \Pr[X = z]\, l_H(z) = 2.6$

$$\Rightarrow \text{We save } 1 - \frac{\mathrm{E}[l_H(z)]}{\mathrm{E}[l_U(z)]} \approx 13\,\%$$

Data compression

Huffman-Code

Notes

- Static Huffman codes rely on the probability of occurrence of characters to match expectations.
- Character frequencies can be determined dynamically, but the recipient must then be informed of the codebook used.
- The Huffman code is an optimal and prefix-free code.

Definition (optimal prefix code)
In a prefix-free code, valid codewords are never true prefixes of another codeword of the same code. Furthermore, an optimal prefix-free code minimizes the mean codeword length

$$\sum_{i \in \mathcal{A}} p(i) \cdot |c(i)|,$$

where $p(i)$ denotes the occurrence probability of $i \in \mathcal{A}$ and $c(i)$ denotes the mapping to a corresponding codeword.

- It is also a variable-length code, i. e., codewords have different lengths.
- Huffman coding is one of the so-called entropy encoding approaches.
- Huffman coding can be applied not only for single characters but also for character strings. However, longer code words can lead to problems as a result of the complexity for determining the codebook.

Data compression

Family of run-length codes

Basic idea

- Data often has repetitions of single characters or groups of characters.
- Instead of encoding each character or group of characters again for repetitions, this is done only once.
- For reconstruction, the number of repetitions is encoded at the affected location.
- Whether individual bits, characters, or entire sequences are encoded depends on the code in question.

Properties

- Lossless compression
- Easy to implement, even in hardware
- In general not optimal

Usage

- Different algorithms for image compression
- Fax
- Analog and ISDN modems

Chapter 5: Session, Presentation, and Application Layer

Application Layer

The application layer is the top layer of the 7 layers of the ISO/OSI reference model. It is the highest interface between applications and the network. Application layer protocols provide specific services.

Examples:

- Domain Name System (DNS) – Resolves fully qualified domain names (FQDNs)[1] to IP addresses and vice versa.
- Hyper Text Transfer Protocol (HTTP) – Used to transfer websites and data.
- File Transfer Protocol (FTP) – Legacy protocol for data transfer between client and server.
- Simple Mail Transfer Protocol (SMTP) – Used to transfer emails between message transfer agents (from client to server and between servers).
- Post Office Protocol (POP) and Internet Message Access Protocol (IMAP) – Protocols to retrieve emails using a mail user agent (email application).
- Telnet – Simple and unsecure protocol for interactive communication with another host.
- Secure Shell (SSH) – Secure interactive access and communication with another host.
- Simple Network Management Protocol (SNMP) – Monitoring ob network components such as switches.

[1] Colloquially referred to as "web addresses"

Domain Name System (DNS)

Motivation:

- When users want to address a computer, e. g. while accessing a website, they do not want to have to remember an IP address.
- Instead, the target is usually addressed by a hierarchically structured name such as `www.google.com`.

The Domain Name System (DNS) consists of three essential components:

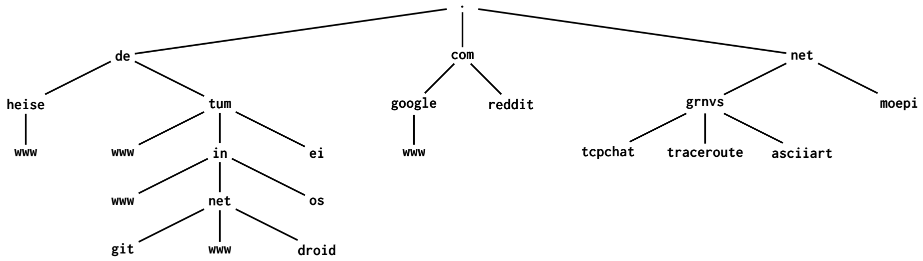
1. The domain namespace
   - is a hierarchically structured namespace with
   - a tree-like structure.
2. Nameserver
   - store information about the namespace where
   - each server is responsible for (small and contiguous) subsections of the namespace.
3. Resolver are applications that
   - extract information from the namespace through queries to nameservers and
   - make it available to requesting clients or applications.

Details: see the respective RFCs [3, 4, 1, 2]

At the same time DNS abstracts from IP addresses, i. e., instead of having to configure the IP address of a server, e. g. in an email application, its name can be specified. The IP address can even change without changing the configuration of the email application.

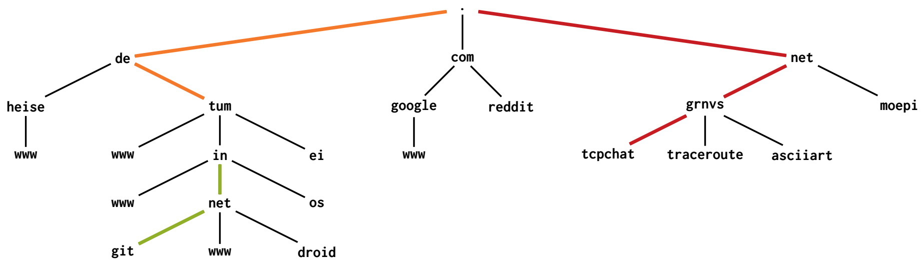
Domain Name System (DNS)

Example



Domain Name System (DNS)

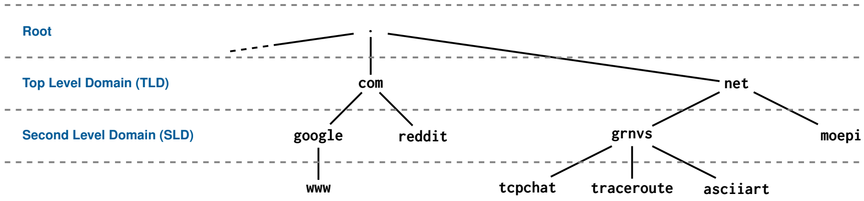
Example



- A label is an arbitrary node in the namespace.
- A domain name is an ordered sequence of labels:
  - A fully qualified domain name (FQDN) consists of the complete sequence of labels originating from a specific node up to the root and ends with a dot, e. g. `tum.de.` or `material.grnvs.net.` .
  - If it does not end with a dot, it is also a domain name. However, its specification is relative starting from a node other than the root, e. g. `git.net.in` .
  - An FQDN may be used as suffix for a not fully qualified name, e. g. `git.net.in` together with `tum.de.` as suffix gives a new FQDN **`git.net.in.tum.de.`** .
  - Whether or not an FQDN exists, i. e., can be resolved to an IP address, remains open at this point.

Domain Name System (DNS)

The first three levels of the namespace hierarchy have their own names:

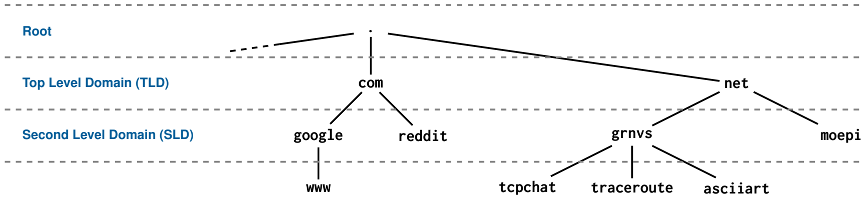


Levels below this are sometimes referred to as subdomains.

[1] Technically, an entry in the DNS could contain all possible octets.

Domain Name System (DNS)

The first three levels of the namespace hierarchy have their own names:



Levels below this are sometimes referred to as subdomains.

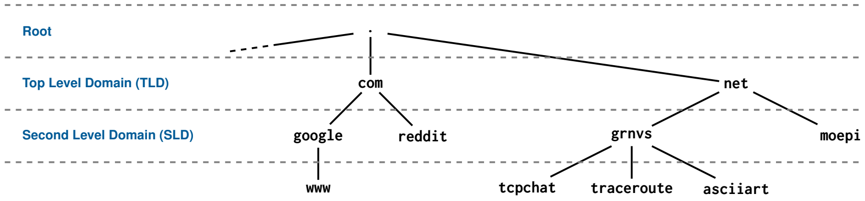
Allocation of TLDs and SLDs

- Top level domains are assigned by the Internet Corporation for Assigned Names and Numbers (ICANN).
- Second level domains are assigned by different registrars.
- In the case of the country code top-level domain (ccTLD) **`.de`**, this is DENIC.

[1] Technically, an entry in the DNS could contain all possible octets.

Domain Name System (DNS)

The first three levels of the namespace hierarchy have their own names:



Levels below this are sometimes referred to as subdomains.

Allocation of TLDs and SLDs

- Top level domains are assigned by the Internet Corporation for Assigned Names and Numbers (ICANN).
- Second level domains are assigned by different registrars.
- In the case of the country code top-level domain (ccTLD) `.de`, this is DENIC.

Character set

- Only letters (`A`–`Z`) and numbers as well as "`-`" are allowed, where the latter must not be the first or last character.[1]
- No distinction is made between upper and lower case.

[1] Technically, an entry in the DNS could contain all possible octets.

Domain Name System (DNS)

Nameserver

The namespace is stored

- as a kind of distributed database
- by a large number of servers, where
- each (name)server is responsible for only a small part of the namespace.

For this purpose, the namespace is divided into zones:

- Zones are contiguous subtrees of the namespace.
- A zone may thus consist of several levels of the namespace, but not of subtrees without common root node.
- Nameservers are called authoritative for the respective zones that they store.
- The same zone may be stored on multiple nameservers.
- There are mechanisms for zone transfers between multiple authoritative nameservers.
- In most cases, there is a primary nameserver (master) where changes within a zone can be made, and a number of secondary nameservers (slaves) that only store copies of the zone.

Nameservers expect incoming requests on port UDP/TCP 53:

- By default UDP 53 is used.
- Requests larger than 512 B are typically sent to TCP 53.
- Zone transfers are always done on TCP 53.

Domain Name System (DNS)

Resource Records

Information is stored in a zone by resource records:

- SOA record (Start of Authority) is a special type of record defining the root of the zone for which a nameserver is authoritative.
- NS records define the FQDNs of nameservers. It may even link to FQDNs in other zones.
- A records map an FQDN to an IPv4 address.
- AAAA records map an FQDN to an IPv6 address.
- CNAME records define an alias, i. e., an FQDN links to a “canonical name", which is again an FQDN.
- MX records define the FQDN of mailservers for a specific domain. It may link to an FQDN located in a different zone.
- TXT records map an FQDN to a string (text). It can be used for various purposes, e. g. to store public keys or signatures.
- PTR records map an IPv4 or IPv6 address back to an FQDN (counterpart to A and AAAA records).

Notes:

- Multiple A or AAAA records (even from different zones) can be associated with the same IP address.
- For a specific FQDN there is at most one CNAME, which acts as alias. There must not by any further record for the respective FQDN.
- For a specific zone there are commonly more than one NS and MX records for redundancy reasons.

Domain Name System (DNS)

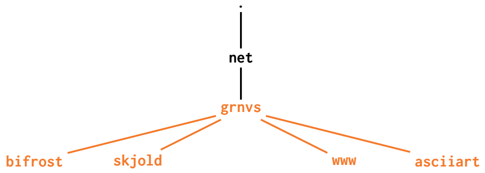
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



Domain Name System (DNS)

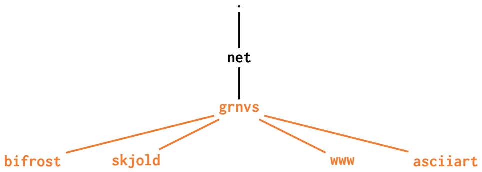
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- Sets the time to live[1] of subsequent resource records to 10 min.

[1] Do not confuse this with the TTL of IP packets.

Domain Name System (DNS)

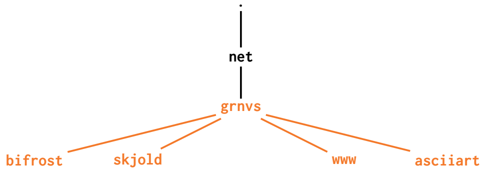
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- SOA record for the domain `grnvs.net.`[2]

[2] In the zone file records are specified as relative domain name (no FQDN) because the starting point was previously specified via `$ORIGIN`.

Domain Name System (DNS)

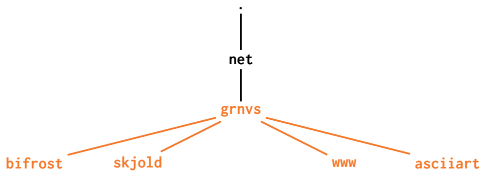
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- SOA record for the domain `grnvs.net.`[2]
  - `serial` denotes the version of the zone file.
  - When the zone file is updated, the serial must be incremented.
  - Used during zone transfers to determine whether secondary servers need to be updated.

[2] In the zone file records are specified as relative domain name (no FQDN) because the starting point was previously specified via `$ORIGIN`.

Domain Name System (DNS)

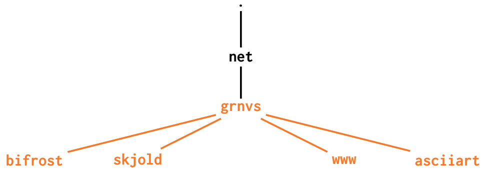
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- SOA record for the domain `grnvs.net.`[2]
  - `refresh` specifies the time interval at which secondaries attempt to match their local copy of the zone file with the primary.

[2] In the zone file records are specified as relative domain name (no FQDN) because the starting point was previously specified via `$ORIGIN`.

Domain Name System (DNS)

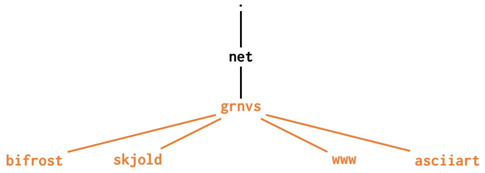
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- SOA record for the domain `grnvs.net.`[2]
  - `retry` specifies the time interval for secondaries to attempt to contact their primary if the previous attempt failed.

[2] In the zone file records are specified as relative domain name (no FQDN) because the starting point was previously specified via `$ORIGIN`.

Domain Name System (DNS)

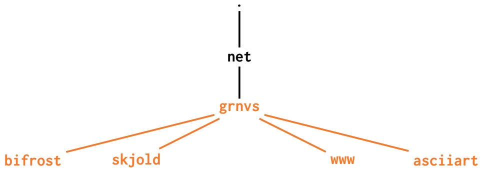
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- SOA record for the domain `grnvs.net.`[2]
  - `expire` specifies for secondaries the maximum time interval since the last successful match with its primary, during which the information of the secondary is considered authoritative.
  - If no contact with the primary is possible within this time, the secondary will stop operating.

[2] In the zone file records are specified as relative domain name (no FQDN) because the starting point was previously specified via `$ORIGIN`.

Domain Name System (DNS)

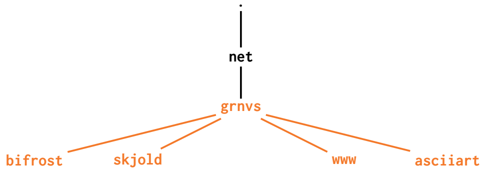
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN   SOA    bifrost.grnvs.net. hostmaster.grnvs.net. (
                           164160 ; serial
                           1800   ; refresh (30 minutes)
                           300    ; retry (5 minutes)
                           604800 ; expire (1 week)
                           1800   ; nxdomain (30 minutes)
                       )
                NS     bifrost.grnvs.net.
                NS     skjold.grnvs.net.
                A      188.95.232.10
                AAAA   2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost         A       95.217.202.138
                AAAA    2a01:4f9:4a:4589::10:1
skjold          A       95.217.202.140
                AAAA    2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart        CNAME   svm0012.net.in.tum.de.
www             A       188.95.232.10
                AAAA    2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- SOA record for the domain `grnvs.net.`[2]
  - `nxdomain`[3] specifies how long requesting resolvers may hold in their cache that a requested resource record does not exist.

[2] In the zone file records are specified as relative domain name (no FQDN) because the starting point was previously specified via `$ORIGIN`.

[3] This field used to have a different meaning.

Domain Name System (DNS)

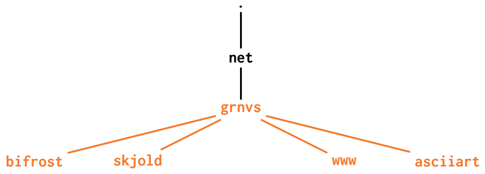
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- The NS records specify the FQDNs of the authoritative nameservers for this zone.
- Their FQDNs do not necessarily have to have the same extension as the current zone.

Domain Name System (DNS)

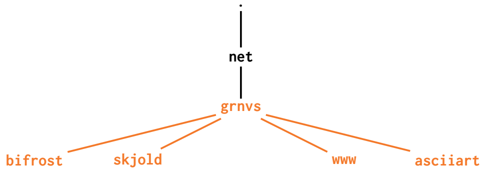
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- A record for the FQDN **grnvs.net.**
- On some domains it is missing and only entries for `www.somedomain.net.` exist. As a result, trying to access `somedomain.net.` in a browser would fail.

Domain Name System (DNS)

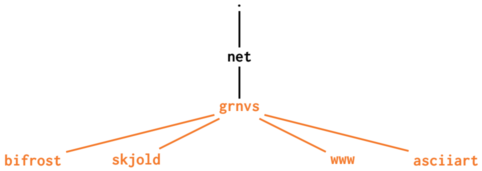
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- Specifies that all subsequent resource records are relative to **$ORIGIN**.
- Abbreviates the remaining resource records, which would all end with **grnvs.net.** otherwise.

Domain Name System (DNS)

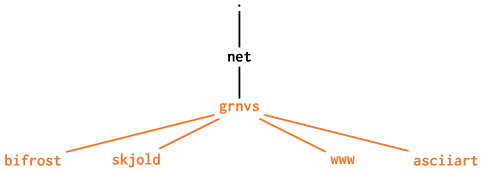
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- A records for three different hosts.
  - The first two are the authoritative nameservers of this zone.
  - The third one is a webserver.

Domain Name System (DNS)

The resource records of a zone are stored on nameservers in the form of zone files:

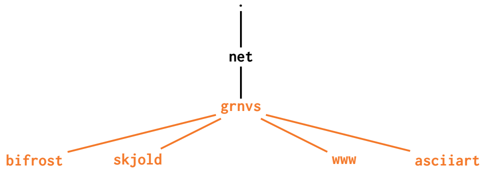
```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

- AAAA records for the hosts (IPv6).

Relevant part of the namespace corresponding to the zone file:



Domain Name System (DNS)

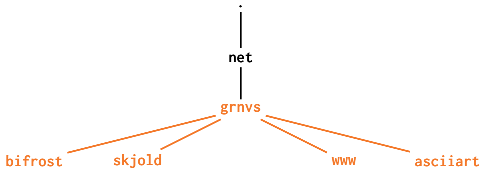
The resource records of a zone are stored on nameservers in the form of zone files:

```
$TTL 600 ; 10 minutes
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    skjold.grnvs.net.
               A     188.95.232.10
               AAAA  2a00:4700:0:9:f::

$ORIGIN grnvs.net.
bifrost        A      95.217.202.138
               AAAA   2a01:4f9:4a:4589::10:1
skjold         A      95.217.202.140
               AAAA   2a01:4f9:4a:4589::20:1

$TTL 60 ; 1 minute
asciiart       CNAME  svm0012.net.in.tum.de.
www            A      188.95.232.10
               AAAA   2a00:4700:0:9:f::
```

Relevant part of the namespace corresponding to the zone file:



- CNAME record that redirects `asciiart.grnvs.net.` to one of our virtual machines.
- The target is not within this zone and must be resolved by separate requests to the respective nameservers of that zone.

Domain Name System (DNS)

Resolver

Resolvers are servers that extract information from the DNS namespace and forward the result to the requesting client.

- Since the DNS corresponds to a distributed database and no single nameserver knows all zones, multiple requests are necessary.
- Resolvers make step-by-step requests to the authoritative nameservers of the respective zones starting at the root.
- The result is returned to the requesting client and can be cached (hopefully respecting the TTL in the SOA record of the respective zone).
- If a client makes the same request again within this time, it can be answered from the cache.

A resolver is not authoritative for any zone.

Problem: how does a resolver know where to start?

Domain Name System (DNS)

Resolver

Resolvers are servers that extract information from the DNS namespace and forward the result to the requesting client.

- Since the DNS corresponds to a distributed database and no single nameserver knows all zones, multiple requests are necessary.
- Resolvers make step-by-step requests to the authoritative nameservers of the respective zones starting at the root.
- The result is returned to the requesting client and can be cached (hopefully respecting the TTL in the SOA record of the respective zone).
- If a client makes the same request again within this time, it can be answered from the cache.

A resolver is not authoritative for any zone.

Problem: how does a resolver know where to start?

- Resolvers have a static list of the 13 root servers[1] that are authoritative for the root zone.
- The root zone is administered by the Internet Corporation for Assigned Names and Numbers (ICANN). However, changes to that zone require the approval of the US Department of Commerce.
- The root servers are operated by various organizations, among others ICANN, Versign, U. S. Army, RIPE, NASA, etc.

[1] In reality, there are hundreds of servers that can be reached via anycast to 13 IP addresses.

Domain Name System (DNS)

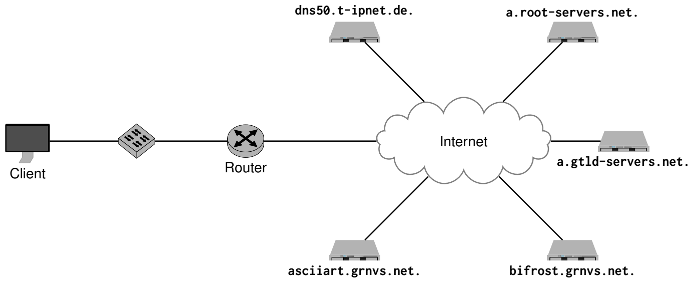
Excerpt from the root hints

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC
;       under anonymous FTP as
;           file                /domain/named.cache
;           on server           FTP.INTERNIC.NET
;       -OR-                    RS.INTERNIC.NET
;
;       last update:    March 23, 2016
;       related version of root zone:   2016032301
;
; formerly NS.INTERNIC.NET
;
.                        3600000      NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
A.ROOT-SERVERS.NET.      3600000      AAAA  2001:503:ba3e::2:30
;
; FORMERLY NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     192.228.79.201
B.ROOT-SERVERS.NET.      3600000      AAAA  2001:500:84::b
;
; FORMERLY C.PSI.NET
;
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
C.ROOT-SERVERS.NET.      3600000      AAAA  2001:500:2::c
;
```

Domain Name System (DNS)

Example: Assume a private Internet connection of a student who accesses the server with FQDN **`asciiart.grnvs.net.`** by means of a web browser.

- The router works as a resolver, but merely forwards any requests to a resolver of the service provider. The IP address of the provider's resolver is known to the router.[1]
- The client uses the router's IP address as resolver.
- Requests of the client to the router as well as from the router to the provider's resolver are recursive queries.
- The actual name resolution is done by the provider's resolver **`dns50.t-ipnet.de.`** through a series of iterative queries to nameservers.

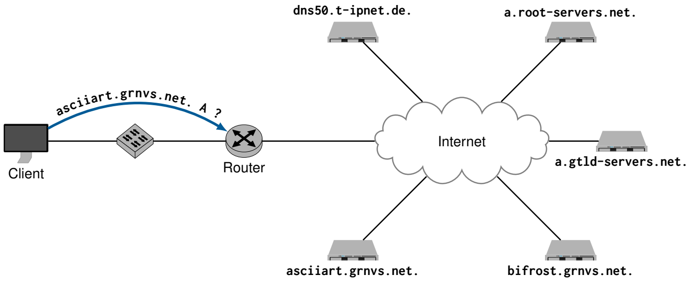


[1] This is known as forwarding. Why this could be beneficial both from the provider's and the user's point of view?

Domain Name System (DNS)

Example: Assume a private Internet connection of a student who accesses the server with FQDN **`asciiart.grnvs.net.`** by means of a web browser.

- The router works as a resolver, but merely forwards any requests to a resolver of the service provider. The IP address of the provider's resolver is known to the router.[1]
- The client uses the router's IP address as resolver.
- Requests of the client to the router as well as from the router to the provider's resolver are recursive queries.
- The actual name resolution is done by the provider's resolver **`dns50.t-ipnet.de.`** through a series of iterative queries to nameservers.

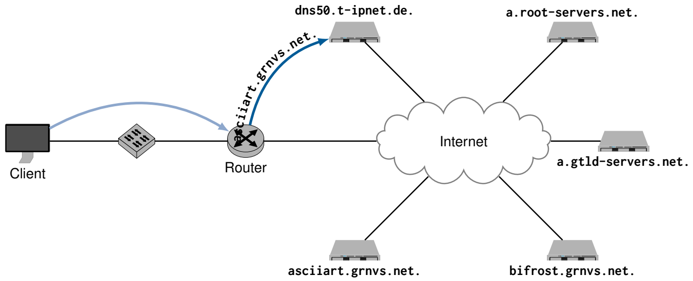


[1] This is known as forwarding. Why this could be beneficial both from the provider's and the user's point of view?

Domain Name System (DNS)

Example: Assume a private Internet connection of a student who accesses the server with FQDN **`asciiart.grnvs.net.`** by means of a web browser.

- The router works as a resolver, but merely forwards any requests to a resolver of the service provider. The IP address of the provider's resolver is known to the router.[1]
- The client uses the router's IP address as resolver.
- Requests of the client to the router as well as from the router to the provider's resolver are recursive queries.
- The actual name resolution is done by the provider's resolver **`dns50.t-ipnet.de.`** through a series of iterative queries to nameservers.

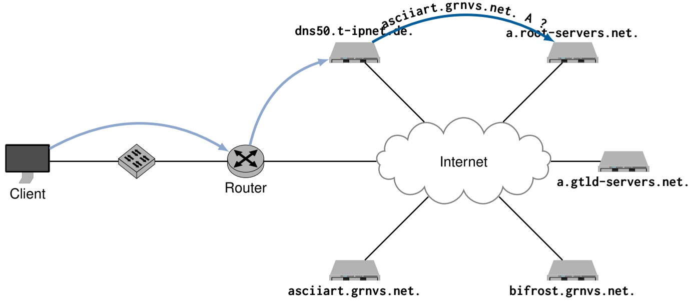


[1] This is known as forwarding. Why this could be beneficial both from the provider's and the user's point of view?

Domain Name System (DNS)

Example: Assume a private Internet connection of a student who accesses the server with FQDN **`asciiart.grnvs.net.`** by means of a web browser.

- The router works as a resolver, but merely forwards any requests to a resolver of the service provider. The IP address of the provider's resolver is known to the router.[1]
- The client uses the router's IP address as resolver.
- Requests of the client to the router as well as from the router to the provider's resolver are recursive queries.
- The actual name resolution is done by the provider's resolver **`dns50.t-ipnet.de.`** through a series of iterative queries to nameservers.

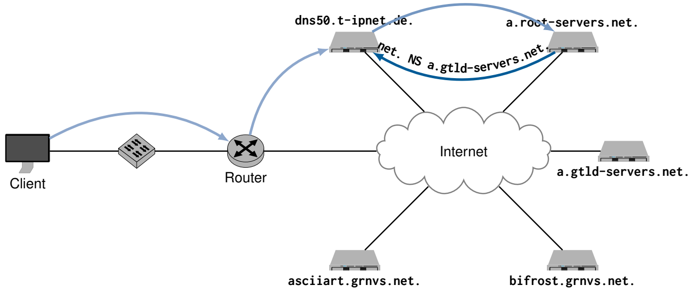


[1] This is known as forwarding. Why this could be beneficial both from the provider's and the user's point of view?

Domain Name System (DNS)

Example: Assume a private Internet connection of a student who accesses the server with FQDN **`asciiart.grnvs.net.`** by means of a web browser.

- The router works as a resolver, but merely forwards any requests to a resolver of the service provider. The IP address of the provider's resolver is known to the router.[1]
- The client uses the router's IP address as resolver.
- Requests of the client to the router as well as from the router to the provider's resolver are recursive queries.
- The actual name resolution is done by the provider's resolver **`dns50.t-ipnet.de.`** through a series of iterative queries to nameservers.

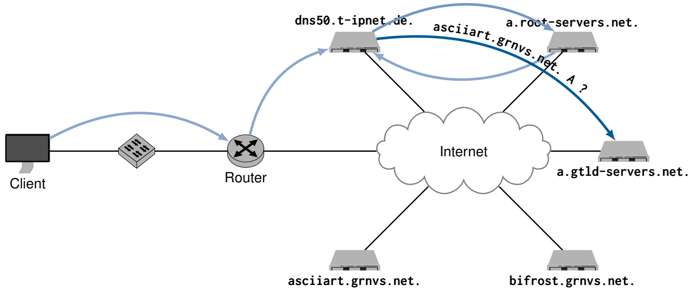


[1] This is known as forwarding. Why this could be beneficial both from the provider's and the user's point of view?

Domain Name System (DNS)

Example: Assume a private Internet connection of a student who accesses the server with FQDN **`asciiart.grnvs.net.`** by means of a web browser.

- The router works as a resolver, but merely forwards any requests to a resolver of the service provider. The IP address of the provider's resolver is known to the router.[1]
- The client uses the router's IP address as resolver.
- Requests of the client to the router as well as from the router to the provider's resolver are recursive queries.
- The actual name resolution is done by the provider's resolver **`dns50.t-ipnet.de.`** through a series of iterative queries to nameservers.

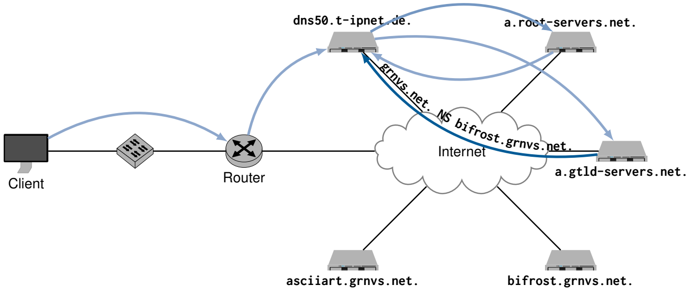


[1] This is known as forwarding. Why this could be beneficial both from the provider's and the user's point of view?

Domain Name System (DNS)

Example: Assume a private Internet connection of a student who accesses the server with FQDN **`asciiart.grnvs.net.`** by means of a web browser.

- The router works as a resolver, but merely forwards any requests to a resolver of the service provider. The IP address of the provider's resolver is known to the router.[1]
- The client uses the router's IP address as resolver.
- Requests of the client to the router as well as from the router to the provider's resolver are recursive queries.
- The actual name resolution is done by the provider's resolver **`dns50.t-ipnet.de.`** through a series of iterative queries to nameservers.

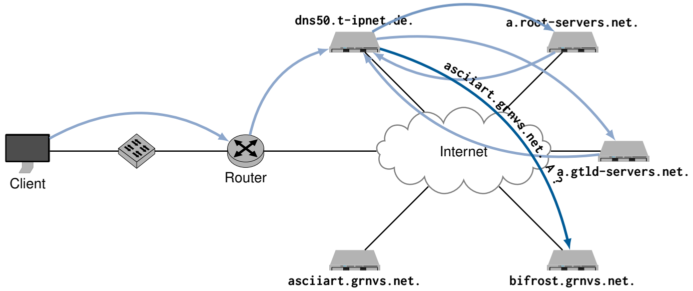


[1] This is known as forwarding. Why this could be beneficial both from the provider's and the user's point of view?

Domain Name System (DNS)

Example: Assume a private Internet connection of a student who accesses the server with FQDN **`asciiart.grnvs.net.`** by means of a web browser.

- The router works as a resolver, but merely forwards any requests to a resolver of the service provider. The IP address of the provider's resolver is known to the router.[1]
- The client uses the router's IP address as resolver.
- Requests of the client to the router as well as from the router to the provider's resolver are recursive queries.
- The actual name resolution is done by the provider's resolver **`dns50.t-ipnet.de.`** through a series of iterative queries to nameservers.

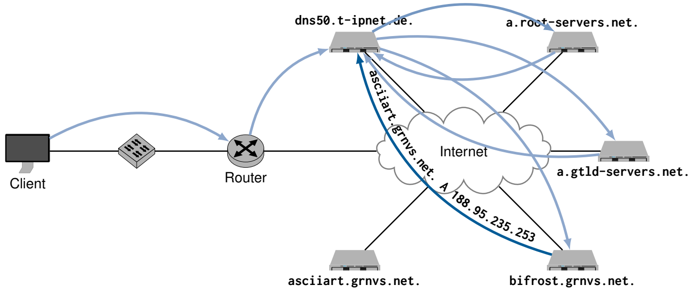


[1] This is known as forwarding. Why this could be beneficial both from the provider's and the user's point of view?

Domain Name System (DNS)

Example: Assume a private Internet connection of a student who accesses the server with FQDN **`asciiart.grnvs.net.`** by means of a web browser.

- The router works as a resolver, but merely forwards any requests to a resolver of the service provider. The IP address of the provider's resolver is known to the router.[1]
- The client uses the router's IP address as resolver.
- Requests of the client to the router as well as from the router to the provider's resolver are recursive queries.
- The actual name resolution is done by the provider's resolver **`dns50.t-ipnet.de.`** through a series of iterative queries to nameservers.

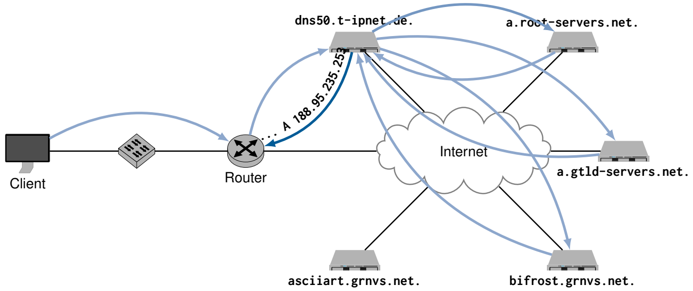


[1] This is known as forwarding. Why this could be beneficial both from the provider's and the user's point of view?

Domain Name System (DNS)

Example: Assume a private Internet connection of a student who accesses the server with FQDN **`asciiart.grnvs.net.`** by means of a web browser.

- The router works as a resolver, but merely forwards any requests to a resolver of the service provider. The IP address of the provider's resolver is known to the router.[1]
- The client uses the router's IP address as resolver.
- Requests of the client to the router as well as from the router to the provider's resolver are recursive queries.
- The actual name resolution is done by the provider's resolver **`dns50.t-ipnet.de.`** through a series of iterative queries to nameservers.

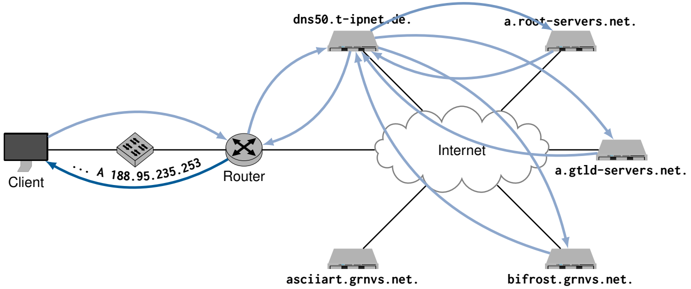


[1] This is known as forwarding. Why this could be beneficial both from the provider's and the user's point of view?

Domain Name System (DNS)

Example: Assume a private Internet connection of a student who accesses the server with FQDN **`asciiart.grnvs.net.`** by means of a web browser.

- The router works as a resolver, but merely forwards any requests to a resolver of the service provider. The IP address of the provider's resolver is known to the router.[1]
- The client uses the router's IP address as resolver.
- Requests of the client to the router as well as from the router to the provider's resolver are recursive queries.
- The actual name resolution is done by the provider's resolver **`dns50.t-ipnet.de.`** through a series of iterative queries to nameservers.



[1] This is known as forwarding. Why this could be beneficial both from the provider's and the user's point of view?

Domain Name System (DNS)

The previous example was simplified in some aspects:

1. The name server **`bifrost.grnvs.net.`** would not have sent an A record with the requested IP address to the resolver, but only a CNAME record.
   - The reason is that there is no A record but only a CNAME record for **`asciiart.grnvs.net.`** according to the zone file on Slide 31.
   - In reality, the resolver would have had to use further queries to determine an authoritative nameserver for **`svm0012.net.in.tum.de.`** (the FQDN provided by the CNAME record).
   - From the point of view of the router and the client, however, nothing would have changed.

2. The name servers' responses to the iterative queries provide only the FQDNs of the authoritative name servers for a zone.
   - So, for example, in the first step the root server **`a.root-servers.net.`** only sends the FQDN of nameservers authoritative for **`net.`** to the resolver.
   - Before the resolver can proceed with its actual task, it must first determine the appropriate A or AAAA record for **`a.gtld-servers.net.`**.
   - If the FQDN of the name server is in the searched zone (as is the case here), so-called glue records must be entered in the zone above it. Glue records contain the IP address of the searched nameservers.
   - From the point of view of the router and client, nothing changes in this case either.

⇒ Resolving an FQDN from scratch therefore require multiple, maybe dozens of requests / responses. Think about what would happen if for each request a TCP connection needs to be created.

Domain Name System (DNS)

Reverse DNS

The namespace can also map FQDNs to IP addresses by using PTR (pointer) records. This is known as reverse DNS. For that purpose dedicated zones for IPv4 and IPv6 exist:

- in-addr.arpa. for IPv4
- ip6.arpa. for IPv6

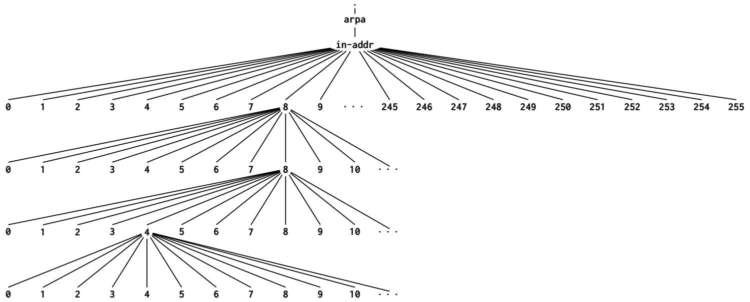
Domain Name System (DNS)

Reverse DNS

The namespace can also map FQDNs to IP addresses by using PTR (pointer) records. This is known as reverse DNS. For that purpose dedicated zones for IPv4 and IPv6 exist:

- in-addr.arpa. for IPv4
- ip6.arpa. for IPv6

The namespace below `in-addr.arpa.` is created by the four octets IPv4 addresses in reverse order (from less specific to more specific):



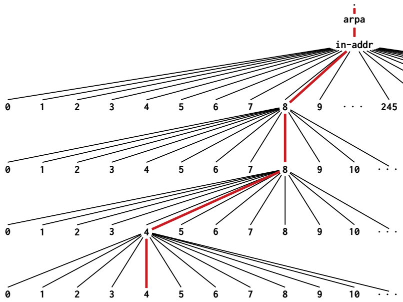
Domain Name System (DNS)

Reverse DNS

The namespace can also map FQDNs to IP addresses by using PTR (pointer) records. This is known as reverse DNS. For that purpose dedicated zones for IPv4 and IPv6 exist:

- in-addr.arpa. for IPv4
- ip6.arpa. for IPv6

The namespace below `in-addr.arpa.` is created by the four octets IPv4 addresses in reverse order (from less specific to more specific):



The IP address **`8.8.4.4`**

- corresponds in reverse DNS to the FQDN **`4.4.8.8.in-addr.arpa.`**, which
- by means of a PTR record
- points to the FQDN **`dns.google.com.`**.

This is one of the two Google resolvers accessible via anycast.

You can check this with Linux or Windows, respectively:
`dig -x 8.8.4.4` or `nslookup 8.8.4.4`

Domain Name System (DNS)

There are some restrictions for the namespace below **`in-addr.arpa.`**:

- Since each level corresponds to a whole octet, there is a maximum of four levels.
- Subnets whose prefix length is not 8, 16, 24, or 32 cannot be stored in separate zones (the latter correspond to individual IP addresses).
- The mapping of subnets of other sizes is possible only with tricks.

Domain Name System (DNS)

There are some restrictions for the namespace below **`in-addr.arpa.`**:

- Since each level corresponds to a whole octet, there is a maximum of four levels.
- Subnets whose prefix length is not 8, 16, 24, or 32 cannot be stored in separate zones (the latter correspond to individual IP addresses).
- The mapping of subnets of other sizes is possible only with tricks.

The namespace for IPv6 (below of **`ip6.arpa.`**) is created in a similar way:

- The division takes place at 4 bit boundaries instead of whole octets.
- This extends the possibilities for dividing into separate zones.
- The namespace is correspondingly larger as a result of the 128 bit IPv6 addresses.

Uniform Resource Locator (URL)

By means of FQDNs, we can

- use DNS to identify the destination of a connection at layer 3,
- bot not specify the application protocol to use nor address specific resource.

[1] If you provide your username and password in this way you could just as easily broadcast your credentials. . .

[2] Within URLs (or. URIs), the terminating . of an FQDN is usually omitted.

Uniform Resource Locator (URL)

By means of FQDNs, we can

- use DNS to identify the destination of a connection at layer 3,
- bot not specify the application protocol to use nor address specific resource.

Uniform Resource Locator (URL) are address specifications of the form

```
<protocol>://[<username>[:<password>]@]<fqdn>[:<port>][/<path>][?<query>][#<fragment>]
```

- `<protocol>` specifies the application protocol, e. g. HTTP(S), FTP, SMTP, etc.
- <username>[:<password>]@ allows the optional specification of a user name and password.[1]
- `<fqdn>` is the fully qualified domain name[2] that identifies the target at layer 3.
- `:<port>` allows the optional specification of a port number different from the well-known port for the application layer protocol.
- `/<path>` allows to specify a path on the target relative to the root `</>` of the directory structure.
- `?<query>` enables the passing of variables in the form `<variable>=<value>`. Variables can be concatenated using `&`.
- `#fragment` allows to reference single fragments or sections in a document.

[1] If you provide your username and password in this way you could just as easily broadcast your credentials...

[2] Within URLs (or. URIs), the terminating `.` of an FQDN is usually omitted.

Uniform Resource Locator (URL)

By means of FQDNs, we can

- use DNS to identify the destination of a connection at layer 3,
- bot not specify the application protocol to use nor address specific resource.

Uniform Resource Locator (URL) are address specifications of the form

```
<protocol>://[<username>[:<password>]@]<fqdn>[:<port>][/<path>][?<query>][#<fragment>]
```

- `<protocol>` specifies the application protocol, e. g. HTTP(S), FTP, SMTP, etc.
- <username>[:<password>]@ allows the optional specification of a user name and password.[1]
- `<fqdn>` is the fully qualified domain name[2] that identifies the target at layer 3.
- `:<port>` allows the optional specification of a port number different from the well-known port for the application layer protocol.
- `/<path>` allows to specify a path on the target relative to the root `</>` of the directory structure.
- `?<query>` enables the passing of variables in the form `<variable>=<value>`. Variables can be concatenated using `&`.
- `#fragment` allows to reference single fragments or sections in a document.

Examples:

- `http://www.tum.de`
- `https://vcs.net.in.tum.de/svn/grnvss15/users/ma92les/exam/IN0010-20150612-0000.pdf`
- `https://www.mymail.alsoinsecure/mybox?user=student&password=nolongeryourscret`

[1] If you provide your username and password in this way you could just as easily broadcast your credentials. . .

[2] Within URLs (or. URIs), the terminating `.` of an FQDN is usually omitted.

Uniform Resource Locator (URL)

But I just type `google.de` and it still works!

It works in most cases for the following reasons:

1. You did type that into your web browser. Your web browser will assume that you most likely want to connect via HTTP(S) and not via FTP or even SMTP (email).
2. Google has set A or AAAA records for `google.de`, so the call will behave the same as with `www.google.de`. This is not to be taken granted: compare `http://in.tum.de` and `http://ei.tum.de`.
3. The domain name, which is not even an FQDN, is silently interpreted as such.

Uniform Resource Locator (URL)

But I just type `google.de` and it still works!

It works in most cases for the following reasons:

1. You did type that into your web browser. Your web browser will assume that you most likely want to connect via HTTP(S) and not via FTP or even SMTP (email).
2. Google has set A or AAAA records for `google.de`, so the call will behave the same as with `www.google.de`. This is not to be taken granted: compare `http://in.tum.de` and `http://ei.tum.de`.
3. The domain name, which is not even an FQDN, is silently interpreted as such.

By the way:

- That it is a “web server” is not decided by the fact that the FQDN is `www.webserver.de`.
- The “web server” might as well be accessible under the FQDN `mail.mydomain.de`.
- Not even the port number decides this:
  - Of course, it is common for a “web server” to be accessible via TCP 80 and a mail server via TCP 25 (well-known ports).
  - However, nothing prevents us from doing anything[1] to make a “web server” reachable on TCP 25 and a mail server reachable on TCP 80.
  - That this makes sense remains to be seen.

[1] Root or administrator privileges are required, otherwise it is not possible to create a listening socket < 1024 under the common operating systems.

HyperText Transfer Protocol (HTTP)

The protocol most commonly used on the Internet to transfer data between client and server is Hyper Text Transfer Protocol (HTTP):

- HTTP defines which requests a client may send and how servers shall respond.
- With one HTTP command at most one “object” (text, graphic, file, etc.) is transferred.
- Commands are interpreted as ASCII encoded text, i. e., it is a text based protocol.
- Incoming HTTP connections are expected on the well-known port TCP 80.
- With HTTP 1.0 the TCP connection is terminated after each request / response pair.

Question: What is the obvious problem with retrieving a web page that is composed of many parts (e. g. a large number of small graphics)?

HyperText Transfer Protocol (HTTP)

The protocol most commonly used on the Internet to transfer data between client and server is Hyper Text Transfer Protocol (HTTP):

- HTTP defines which requests a client may send and how servers shall respond.
- With one HTTP command at most one “object” (text, graphic, file, etc.) is transferred.
- Commands are interpreted as ASCII encoded text, i. e., it is a text based protocol.
- Incoming HTTP connections are expected on the well-known port TCP 80.
- With HTTP 1.0 the TCP connection is terminated after each request / response pair.

Question: What is the obvious problem with retrieving a web page that is composed of many parts (e. g. a large number of small graphics)?

A new TCP connection has to be established for each object, which costs time accordingly and means a corresponding overhead for many small objects.

Improvements with HTTP 1.1:

- TCP connections outlast multiple requests.
- HTTP can still be considered stateless because the individual requests are independent of each other.

HyperText Transfer Protocol (HTTP)

HTTP Message Types and Headers

HTTP basically distinguishes between two message types: request and response.

1. Request (from client to server), contains
   - a method describing the action desired by the client, e. g. transferring a particular resource from the server to the client,
   - path and query parameters of the Uniform Resource Locator (URL), which describes the requested resource in more detail, and
   - a number of other header fields, which may contain (among others) the following information:
     - FQDN of the requested host[1]
     - Character set and encoding in which the response is expected
     - From where a request came (i. e., redirection from another website via the referrer [2])
     - The user agent, i. e., the client software used

[1] This field allows the server to decide which data is to be delivered for so-called virtual hosts, regardless of the IP

[2] In HTTP spelled incorrectly as Referer [sic].

HyperText Transfer Protocol (HTTP)

HTTP Message Types and Headers

HTTP basically distinguishes between two message types: request and response.

1. Request (from client to server), contains
   - a method describing the action desired by the client, e. g. transferring a particular resource from the server to the client,
   - path and query parameters of the Uniform Resource Locator (URL), which describes the requested resource in more detail, and
   - a number of other header fields, which may contain (among others) the following information:
     - FQDN of the requested host[1]
     - Character set and encoding in which the response is expected
     - From where a request came (i. e., redirection from another website via the referrer [2])
     - The user agent, i. e., the client software used

2. Response (from server to client), contains
   - a status line (numeric code + text to indicate errors),
   - a response header, possibly with additional options, and
   - the body separated by `CRLF` (carriage return line feed), which contains the actual data.

---

[1] This field allows the server to decide which data is to be delivered for so-called virtual hosts, regardless of the IP

[2] In HTTP spelled incorrectly as Referer [sic].

HyperText Transfer Protocol (HTTP)

HTTP Methods and Response Codes

Clients use a number of different commands (methods):

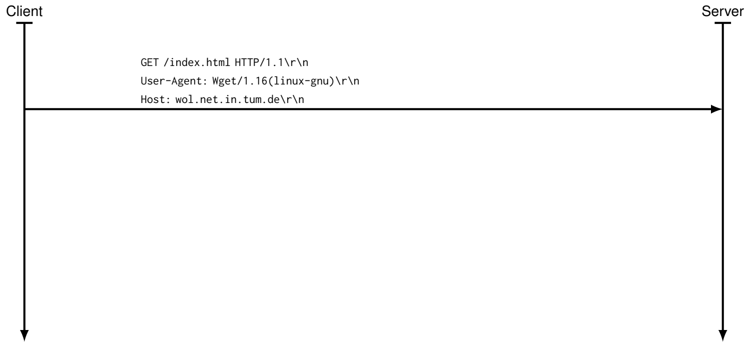
- `GET` — request to transfer a specific object from the server
- `HEAD` — request to transfer the header of a given object (e. g. web pages consist of several sections, one of which is called header)
- `PUT` — transfer of an object from the client to the server, which may overwrite an existing object
- `POST` — transfer of an object from the client to the server, which may be appended to an existing object (e. g. appending text)
- `DELETE` — delete an object from the server

The responses of the server indicate the result of the request by means of the status line, e. g.

- `200` — ok
- `3xx` — redirection
- `400` — bad request
- `401` — unauthorized
- `403` — forbidden
- `404` — not found
- `418` — I'm a teapot (RFC 2324)
- `5xx` — server error
- etc.

HyperText Transfer Protocol (HTTP)

Example: accessing `http://wol.net.in.tum.de/index.html`



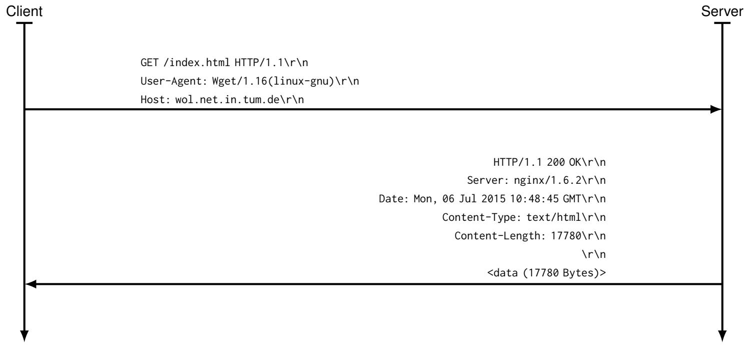
```
GET /index.html HTTP/1.1\r\n
User-Agent: Wget/1.16(linux-gnu)\r\n
Host: wol.net.in.tum.de\r\n
```

- Client uses `GET` to request a specific resource relative to the document root.
- Among other things, the client sends the user agent used, i. e., the browser used.
- `Host` once again explicitly specifies the FQDN of the web server.[1]

[1] At first glance, this is superfluous, since this has already been resolved into an IP address using DNS and thus identifies the target at the network layer. However, resending allows multiple web servers with different FQDNs or URLs to be accessible under the same IP address and port number.

HyperText Transfer Protocol (HTTP)

Example: accessing `http://wol.net.in.tum.de/index.html`



```
GET /index.html HTTP/1.1\r\n
User-Agent: Wget/1.16(linux-gnu)\r\n
Host: wol.net.in.tum.de\r\n
```

```
                    HTTP/1.1 200 OK\r\n
                 Server: nginx/1.6.2\r\n
Date: Mon, 06 Jul 2015 10:48:45 GMT\r\n
             Content-Type: text/html\r\n
              Content-Length: 17780\r\n
                                  \r\n
                   <data (17780 Bytes)>
```

- The server responds with the appropriate status code.
- In addition, various options are specified[2], here in particular the content type. Often the content encoding and character set are also specified.
- The (appropriately coded) data follows at the end.

[2] Due to space limitations, not all options are shown.

HyperText Transfer Protocol (HTTP)

Encryption with HTTP

- HTTP itself does not provide any mechanisms for encrypting transmitted data.
- However, it is possible to use appropriate protocols between the transport and application layers.

Hyper Text Transfer Protocol Secure (HTTPS)

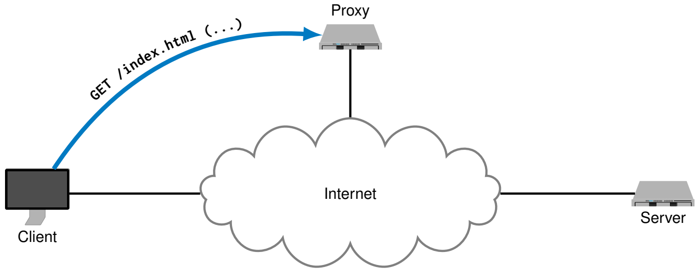
- Uses Transport Layer Security (TLS), an encryption protocol above the transport layer.
- TLS encrypts and decrypts the data transfer.
- HTTP itself remains unchanged.
- TCP 443 is used instead of TCP 80 to distinguish from unencrypted connections.
- For TLS, HTTP is just a data stream.

HyperText Transfer Protocol (HTTP)

HTTP Proxy

Sometimes it is useful or necessary for a client not to establish an end-to-end connection to an HTTP server, but to use a proxy:

- Instead of contacting the destination server directly, HTTP requests are sent to an HTTP proxy.
- The proxy accepts the requests, and in turn establishes a new connection to the actual target server (or another proxy), making the request on behalf of the client.
- The corresponding response is first sent to the proxy, which delivers it (hopefully unmodified) to the client.
- To distinguish them from normal HTTP servers, proxies often use different port numbers such as TCP 3128.

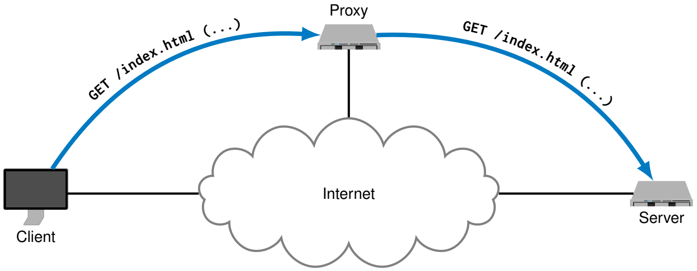


HyperText Transfer Protocol (HTTP)

HTTP Proxy

Sometimes it is useful or necessary for a client not to establish an end-to-end connection to an HTTP server, but to use a proxy:

- Instead of contacting the destination server directly, HTTP requests are sent to an HTTP proxy.
- The proxy accepts the requests, and in turn establishes a new connection to the actual target server (or another proxy), making the request on behalf of the client.
- The corresponding response is first sent to the proxy, which delivers it (hopefully unmodified) to the client.
- To distinguish them from normal HTTP servers, proxies often use different port numbers such as TCP 3128.

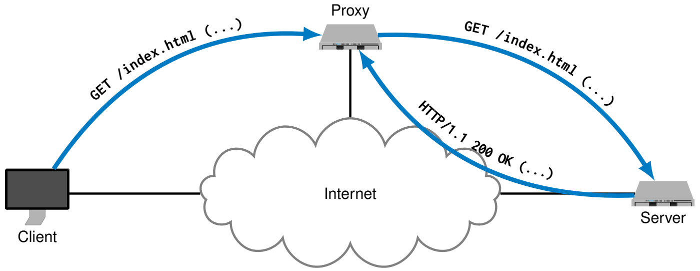


HyperText Transfer Protocol (HTTP)

HTTP Proxy

Sometimes it is useful or necessary for a client not to establish an end-to-end connection to an HTTP server, but to use a proxy:

- Instead of contacting the destination server directly, HTTP requests are sent to an HTTP proxy.
- The proxy accepts the requests, and in turn establishes a new connection to the actual target server (or another proxy), making the request on behalf of the client.
- The corresponding response is first sent to the proxy, which delivers it (hopefully unmodified) to the client.
- To distinguish them from normal HTTP servers, proxies often use different port numbers such as TCP 3128.

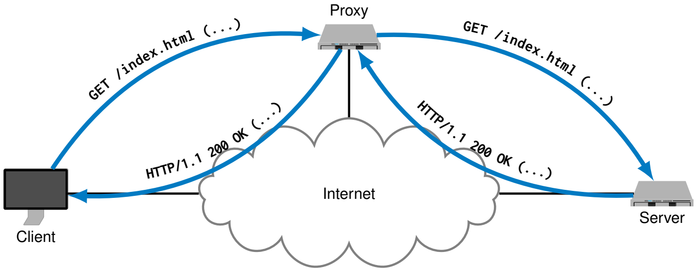


HyperText Transfer Protocol (HTTP)

HTTP Proxy

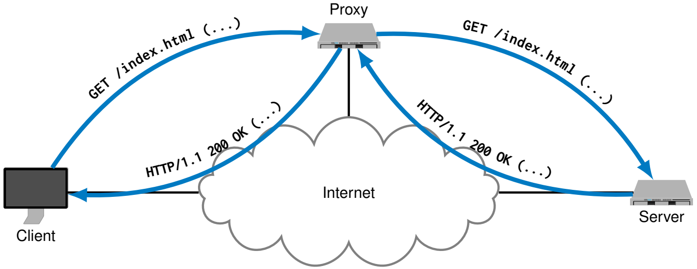
Sometimes it is useful or necessary for a client not to establish an end-to-end connection to an HTTP server, but to use a proxy:

- Instead of contacting the destination server directly, HTTP requests are sent to an HTTP proxy.
- The proxy accepts the requests, and in turn establishes a new connection to the actual target server (or another proxy), making the request on behalf of the client.
- The corresponding response is first sent to the proxy, which delivers it (hopefully unmodified) to the client.
- To distinguish them from normal HTTP servers, proxies often use different port numbers such as TCP 3128.



HyperText Transfer Protocol (HTTP)

HTTP Proxy



- The client itself remains (largely) hidden behind the proxy, i. e., the web server only sees the proxy but not the client.[1]
- The proxy can cache requests, e. g. in case of multiple identical requests (from different clients) content can be delivered from a cache.
  - Is the content still up-to-date?
  - How and when does content need to be updated?
- Proxies can also operate transparently, i. e., without the clients' knowledge.
  - Commonly found in corporate networks.
  - In some cases, proxies can even be "abused" to monitor TLS encrypted connections.[2]

[1] Of course, this does not affect information that the client sends to the server using `POST`.

[2] This assumes that clients trust the proxy's SSL certificate, which is easy to do in the context of administered networks.

Simple Mail Transfer Protocol (SMTP)

The Simple Mail Transfer Protocol (SMTP) is a text-based protocol to

- send emails, i. e., transporting emails from the mail user agent (MUA) to a mail transfer agent (MTA)[1], and to
- forward emails between MTAs.

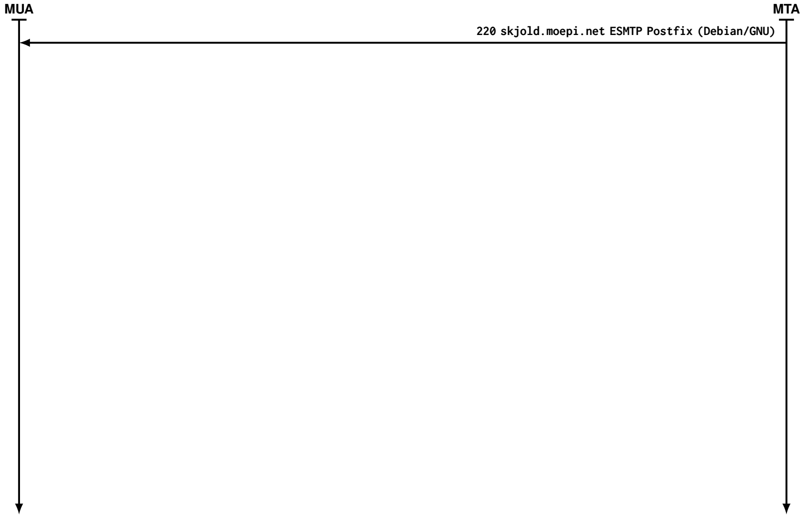
In order to receive emails, on of the protocols

- Post Office Protocol (POP) or
- Internet Message Access Protocol (IMAP) are used.

[1] An email client (applications such as Thunderbird) consists in many cases of a combination of a MUA together with a simple MTA. The latter allows to queue emails for transmission even if no MTA is currently available, e. g. offline use. As soon as a connection can be established (commonly the preconfigured SMTP server of your email provider), the mail is actually transmitted.

Simple Mail Transfer Protocol (SMTP)

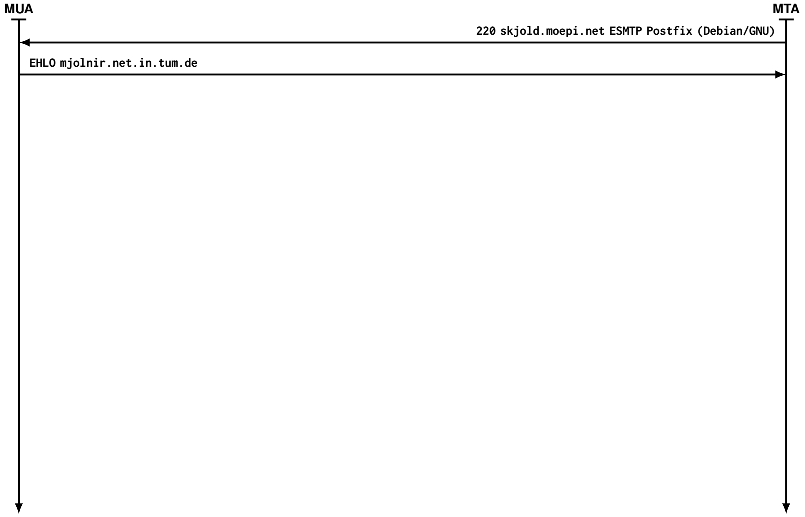
Example: SMTP



```
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
```

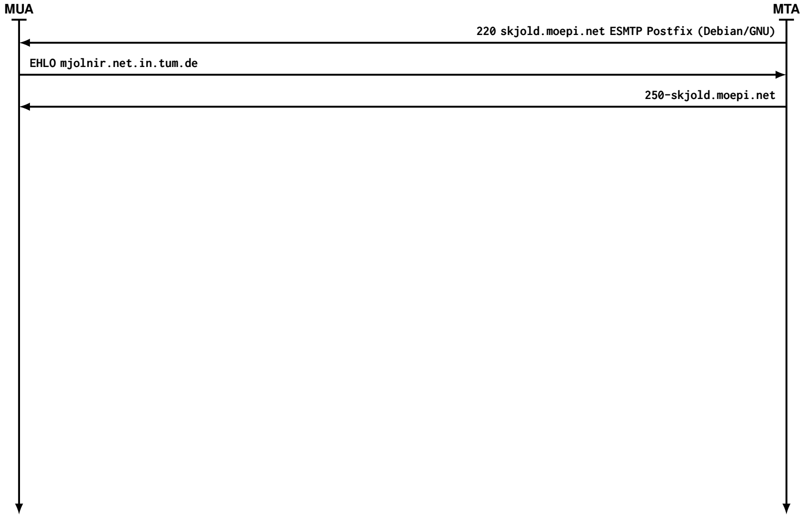
Simple Mail Transfer Protocol (SMTP)

Example: SMTP


MUA
MTA
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de

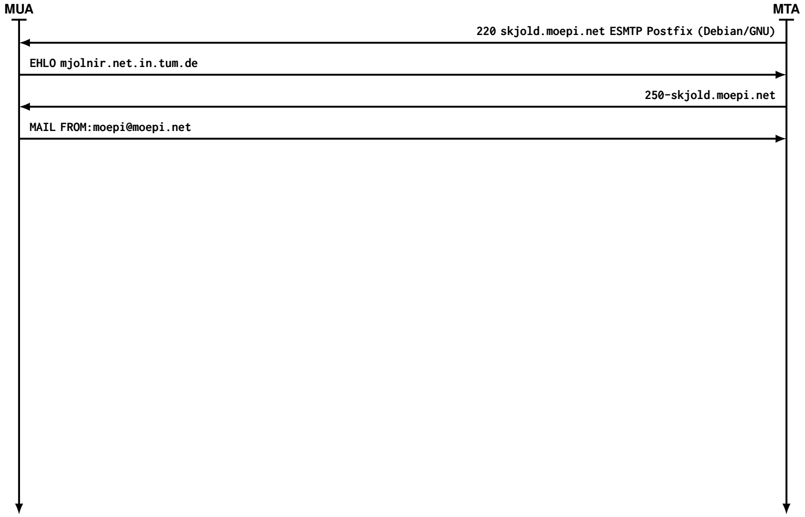
Simple Mail Transfer Protocol (SMTP)

Example: SMTP


MUA
MTA
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de
250-skjold.moepi.net

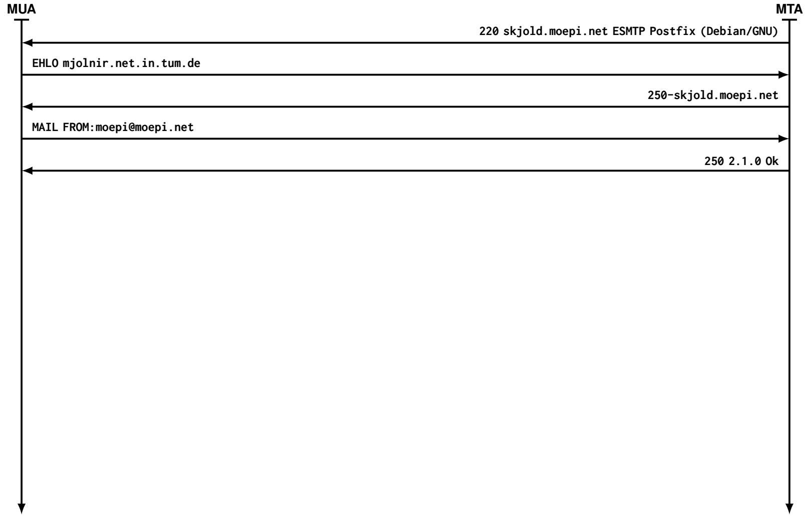
Simple Mail Transfer Protocol (SMTP)

Example: SMTP


MUA
MTA
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de
250-skjold.moepi.net
MAIL FROM:moepi@moepi.net

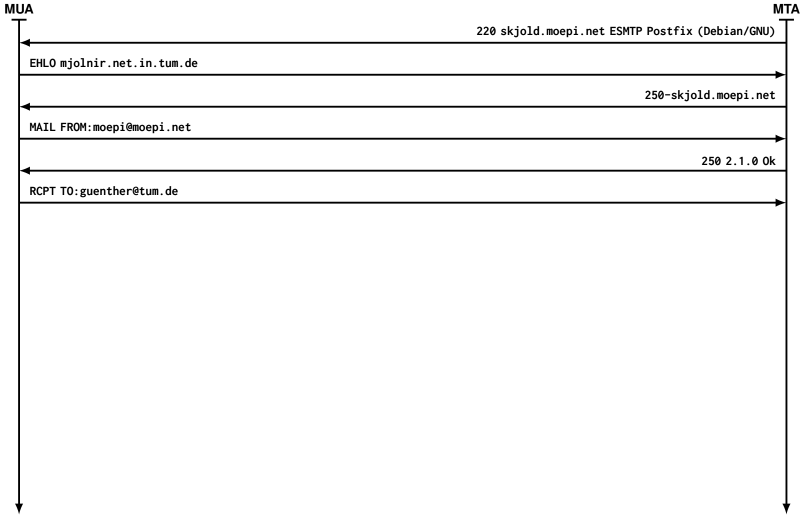
Simple Mail Transfer Protocol (SMTP)

Example: SMTP


MUA
MTA
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de
250-skjold.moepi.net
MAIL FROM:moepi@moepi.net
250 2.1.0 Ok

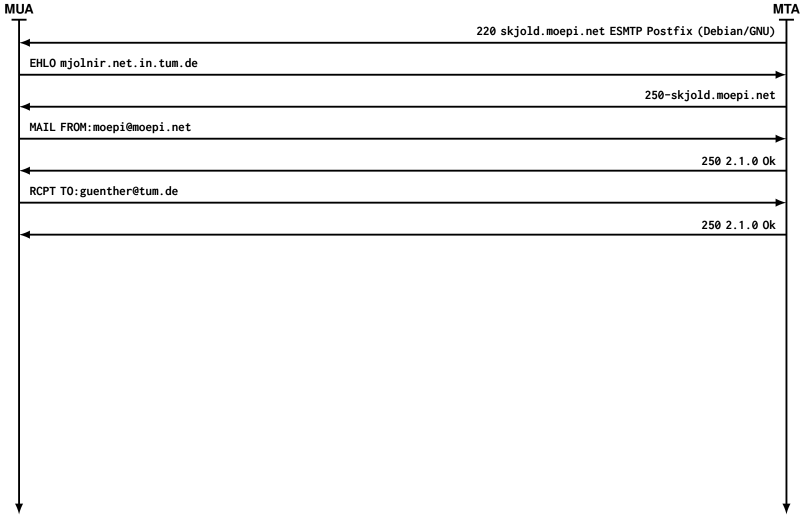
Simple Mail Transfer Protocol (SMTP)

Example: SMTP


MUA
MTA
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de
250-skjold.moepi.net
MAIL FROM:moepi@moepi.net
250 2.1.0 Ok
RCPT TO:guenther@tum.de

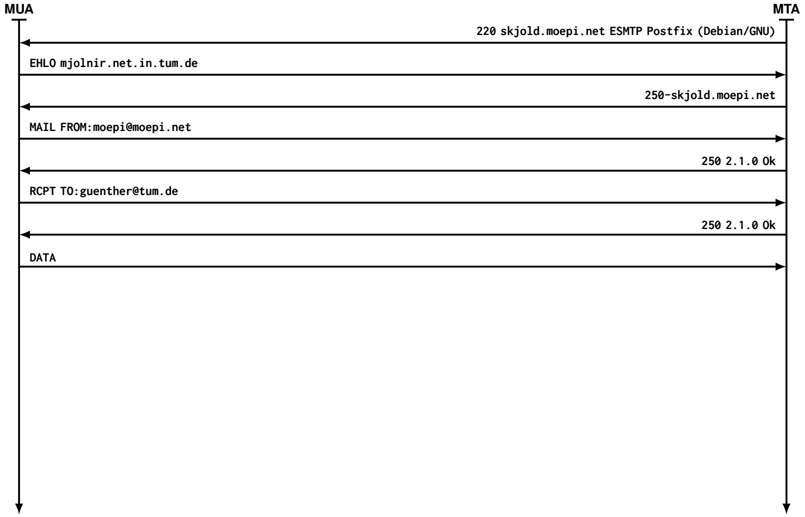
Simple Mail Transfer Protocol (SMTP)

Example: SMTP


MUA
MTA
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de
250-skjold.moepi.net
MAIL FROM:moepi@moepi.net
250 2.1.0 Ok
RCPT TO:guenther@tum.de
250 2.1.0 Ok

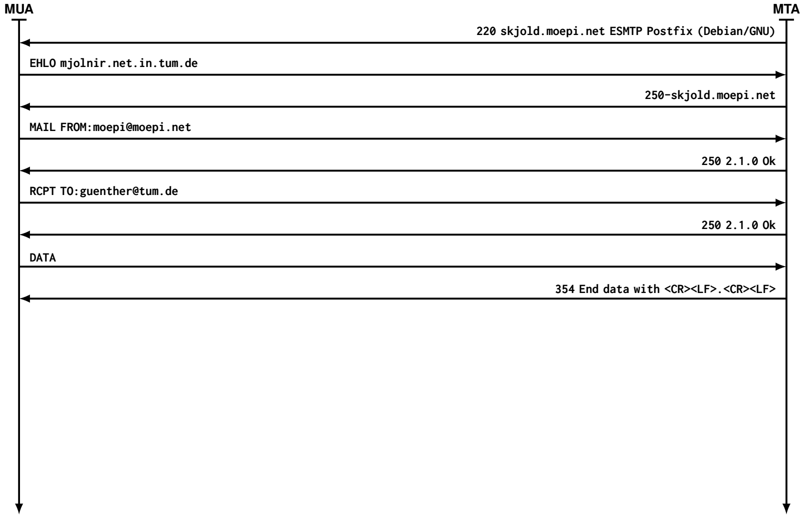
Simple Mail Transfer Protocol (SMTP)

Example: SMTP


MUA
MTA
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de
250-skjold.moepi.net
MAIL FROM:moepi@moepi.net
250 2.1.0 Ok
RCPT TO:guenther@tum.de
250 2.1.0 Ok
DATA

Simple Mail Transfer Protocol (SMTP)

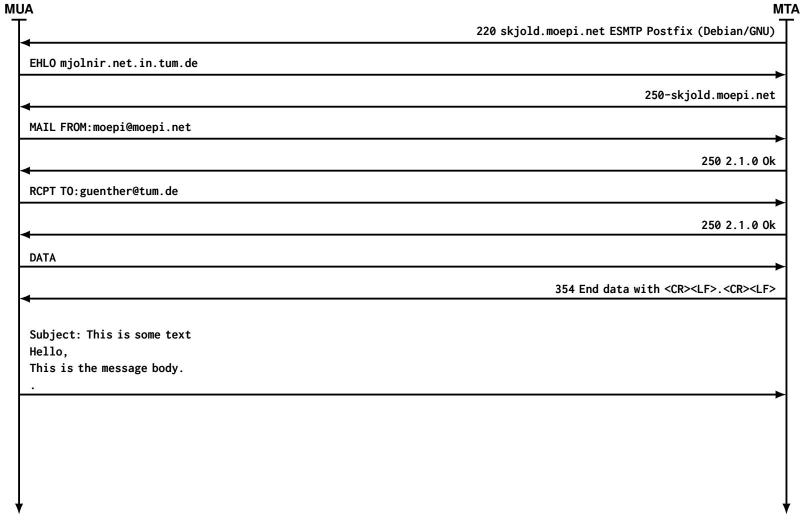
Example: SMTP



MUA
MTA
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de
250-skjold.moepi.net
MAIL FROM:moepi@moepi.net
250 2.1.0 Ok
RCPT TO:guenther@tum.de
250 2.1.0 Ok
DATA
354 End data with <CR><LF>.<CR><LF>

Simple Mail Transfer Protocol (SMTP)

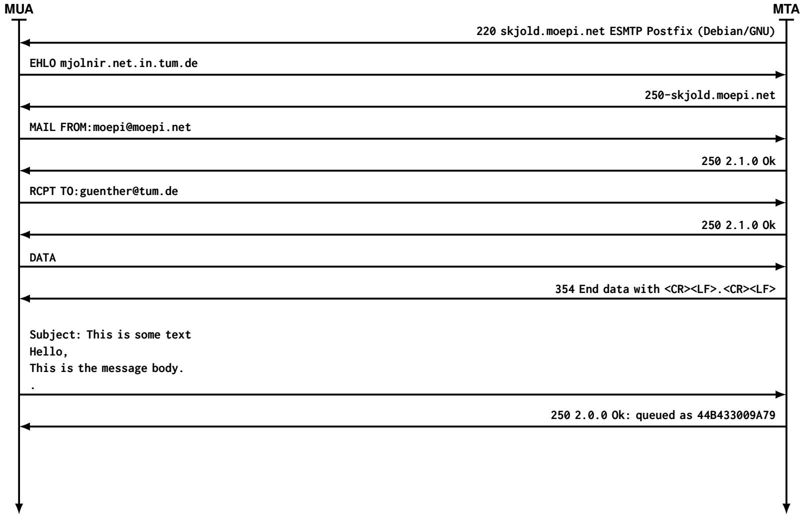
Example: SMTP



```
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de
250-skjold.moepi.net
MAIL FROM:moepi@moepi.net
250 2.1.0 Ok
RCPT TO:guenther@tum.de
250 2.1.0 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: This is some text
Hello,
This is the message body.
.
```

Simple Mail Transfer Protocol (SMTP)

Example: SMTP

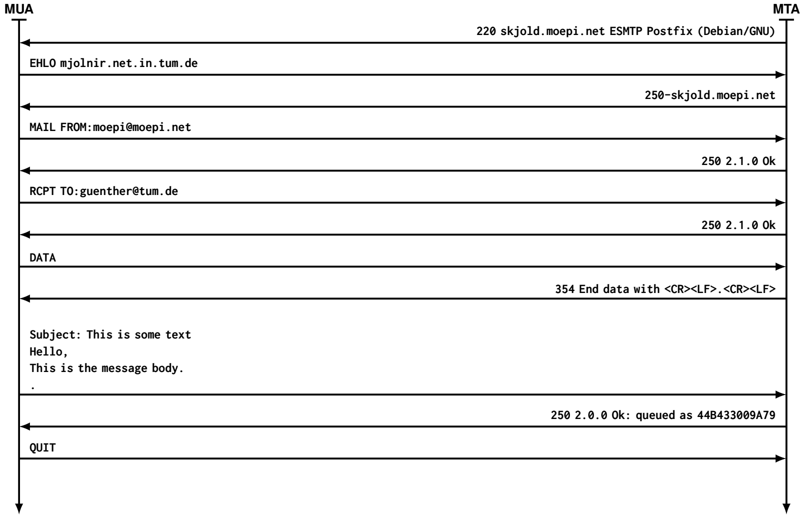


```
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de
250-skjold.moepi.net
MAIL FROM:moepi@moepi.net
250 2.1.0 Ok
RCPT TO:guenther@tum.de
250 2.1.0 Ok
DATA
354 End data with <CR><LF>.<CR><LF>

Subject: This is some text
Hello,
This is the message body.
.
250 2.0.0 Ok: queued as 44B433009A79
```

Simple Mail Transfer Protocol (SMTP)

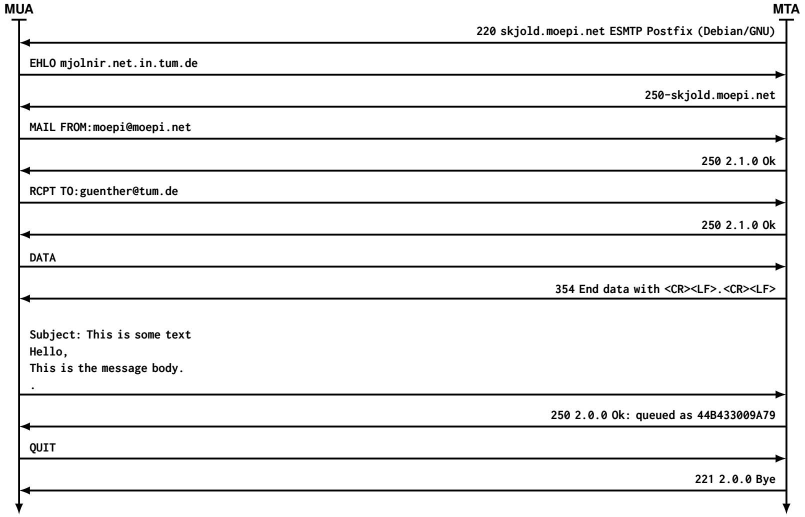
Example: SMTP



```
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de
250-skjold.moepi.net
MAIL FROM:moepi@moepi.net
250 2.1.0 Ok
RCPT TO:guenther@tum.de
250 2.1.0 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: This is some text
Hello,
This is the message body.
.
250 2.0.0 Ok: queued as 44B433009A79
QUIT
```

Simple Mail Transfer Protocol (SMTP)

Example: SMTP



```
220 skjold.moepi.net ESMTP Postfix (Debian/GNU)
EHLO mjolnir.net.in.tum.de
250-skjold.moepi.net
MAIL FROM:moepi@moepi.net
250 2.1.0 Ok
RCPT TO:guenther@tum.de
250 2.1.0 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: This is some text
Hello,
This is the message body.
.
250 2.0.0 Ok: queued as 44B433009A79
QUIT
221 2.0.0 Bye
```

Simple Mail Transfer Protocol (SMTP)

Example: SMTP

When an MTA receives an email, it tries to deliver it to the correct destination:

- The recipient's address `guenther@tum.de` contains the FQDN of the target domain.
- Using DNS the MX records and thus the correct MTAs for `tum.de` are identified:

  ```
  tum.de.  3600  IN  MX  100  postrelay2.lrz.de.
  tum.de.  3600  IN  MX  100  postrelay1.lrz.de.
  ```

  The records contain a TTL, the type of the DNS record, a preference (smaller values correspond to higher preference), and the FQDN of the respective MTA.
- The MTA now tries to establish an SMTP connection to the target MTA in order to forward the email.
  - If successful, the message is removed from the transmitting MTA. The sender is not notified.
  - If the target MTA is unavailable, the transmitting MTA will retry in regular intervals. After some time a delay notification may returned to the sender. If the retries remain unsuccessful, the MTA may eventually give up and notify the sender.
  - If the target MTA refuses to accept the message, the sender is notified with the respective error message received from the target.

Simple Mail Transfer Protocol (SMTP)

Example: SMTP

When an MTA receives an email, it tries to deliver it to the correct destination:

- The recipient's address `guenther@tum.de` contains the FQDN of the target domain.
- Using DNS the MX records and thus the correct MTAs for `tum.de` are identified:

```
tum.de.  3600  IN  MX  100  postrelay2.lrz.de.
tum.de.  3600  IN  MX  100  postrelay1.lrz.de.
```

  The records contain a TTL, the type of the DNS record, a preference (smaller values correspond to higher preference), and the FQDN of the respective MTA.
- The MTA now tries to establish an SMTP connection to the target MTA in order to forward the email.
  - If successful, the message is removed from the transmitting MTA. The sender is not notified.
  - If the target MTA is unavailable, the transmitting MTA will retry in regular intervals. After some time a delay notification may returned to the sender. If the retries remain unsuccessful, the MTA may eventually give up and notify the sender.
  - If the target MTA refuses to accept the message, the sender is notified with the respective error message received from the target.

The email is delivered to inbox of the recipient:

- POP3 allows to fetch the message from the server. The message is normally deleted from the server and stored locally in the MUA of the recipient.
- IMAP4 allows to synchronize a local copy of the mailbox with the server, which also allows access to the same content from multiple clients.

Simple Mail Transfer Protocol (SMTP)

Email (Example):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de  Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
  by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
  (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
  (Client did not present a certificate)
  by postrelay1.lrz.de (Postfix) with ESMTPS
  for <guenther@tum.de>; Wed,  1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
  by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
  Wed,  1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
 <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
 Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Example):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de  Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
  by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
  (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
  (Client did not present a certificate)
  by postrelay1.lrz.de (Postfix) with ESMTPS
  for <guenther@tum.de>; Wed,  1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
  by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
  Wed,  1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
 <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
 Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Example):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de  Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
  by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
  (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
  (Client did not present a certificate)
  by postrelay1.lrz.de (Postfix) with ESMTPS
  for <guenther@tum.de>; Wed,  1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
  by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
  Wed,  1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
 <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
 Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Example):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de  Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
  by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
  (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
  (Client did not present a certificate)
  by postrelay1.lrz.de (Postfix) with ESMTPS
  for <guenther@tum.de>; Wed,  1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
  by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
  Wed,  1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
 <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
 Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Example):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de  Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
  by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
  (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
  (Client did not present a certificate)
  by postrelay1.lrz.de (Postfix) with ESMTPS
  for <guenther@tum.de>; Wed,  1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
  by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
  Wed,  1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
 <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
 Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Example):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de  Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
  by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
  (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
  (Client did not present a certificate)
  by postrelay1.lrz.de (Postfix) with ESMTPS
  for <guenther@tum.de>; Wed,  1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
  by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
  Wed,  1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
 <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
 Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Example):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de  Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
  by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
  (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
  (Client did not present a certificate)
  by postrelay1.lrz.de (Postfix) with ESMTPS
  for <guenther@tum.de>; Wed,  1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
  by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
  Wed,  1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
 <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
 Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Example):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de  Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
  by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
  (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
  (Client did not present a certificate)
  by postrelay1.lrz.de (Postfix) with ESMTPS
  for <guenther@tum.de>; Wed,  1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
  by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
  Wed,  1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
 <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
 Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Example):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de  Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
  by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
  (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
  (Client did not present a certificate)
  by postrelay1.lrz.de (Postfix) with ESMTPS
  for <guenther@tum.de>; Wed,  1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
  by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
  Wed,  1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
 <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
 Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Example):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de  Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
  by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
  (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
  (Client did not present a certificate)
  by postrelay1.lrz.de (Postfix) with ESMTPS
  for <guenther@tum.de>; Wed,  1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
  by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
  Wed,  1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
 <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
 Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Example):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de  Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
  by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
  (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
  (Client did not present a certificate)
  by postrelay1.lrz.de (Postfix) with ESMTPS
  for <guenther@tum.de>; Wed,  1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
  by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
  Wed,  1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
 <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
 Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)
```

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Notes:

- In general, an MTA only accepts emails for its own domains but not for others. Otherwise one talks about an open relay, which would be abused to send spam messages and eventually end up on blacklists.
- Authentication between MTAs is unusual as it is difficult to realize.[1]
- The messages exchange between MTAs may still be encrypted. This is called opportunistic encryption since MTAs are in general not required to support it. Note that there is no way for authentication, only the email transport is encrypted. Users do not have any influence on that.
- Both SMTP and POP / IMAP can be encrypted using TLS. The port numbers change accordingly (cmp. HTTP and HTTPS).

[1] An exception are MTAs that act as smarthost or emphrelay host. Such an MTA accepts emails from known MTAs after proper authentication since these would otherwise not be able to successfully send emails on their own. Examples are MTAs with dynamic IP address, which in most cases do not have valid PTR records and may thus be rejected by other MTAs.

File Transfer Protocol (FTP)

The File Transfer Protocol (FTP) [5] is another protocol to transfer data (text and binary data). It is considered a legacy protocol in most browsers, but has some interesting mechanisms. Differences to HTTP:

- FTP uses two separate TCP connections:
    1. a control channel to exchange commands and status codes between client and server
    2. a data channel for file transfer
- The control channel remains established for multiple file transfers, i. e., FTP is stateful and also implements functions of the session layer.
- FTP requires authentication (anonymous access may be granted by the user name `anonymous` and an arbitrary password).

File Transfer Protocol (FTP)

The File Transfer Protocol (FTP) [5] is another protocol to transfer data (text and binary data). It is considered a legacy protocol in most browsers, but has some interesting mechanisms. Differences to HTTP:

- FTP uses two separate TCP connections:
    1. a control channel to exchange commands and status codes between client and server
    2. a data channel for file transfer
- The control channel remains established for multiple file transfers, i. e., FTP is stateful and also implements functions of the session layer.
- FTP requires authentication (anonymous access may be granted by the user name `anonymous` and an arbitrary password).

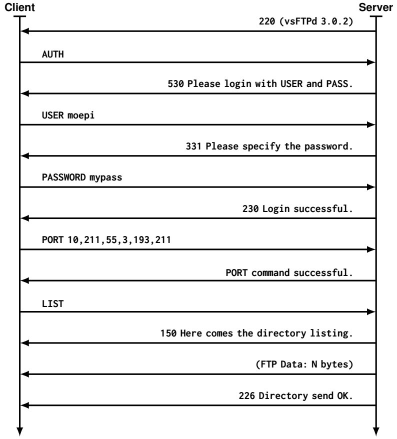
FTP works either in active (default) or passive mode:

- In both cases the client establishes a control channel to the server on port TCP 21.
- In active mode, the client uses the `PORT` to send a random ephemeral port to the server. The server then establishes a new TCP connection with source port TCP 20 to the client on that port. This connection is used as data channel.
- In passive mode, the client first signals its use via the `PASV`. The server send sends IP address and port number to which the client should establish a new connection for the data channel.

File Transfer Protocol (FTP)

Example:

FTP active mode

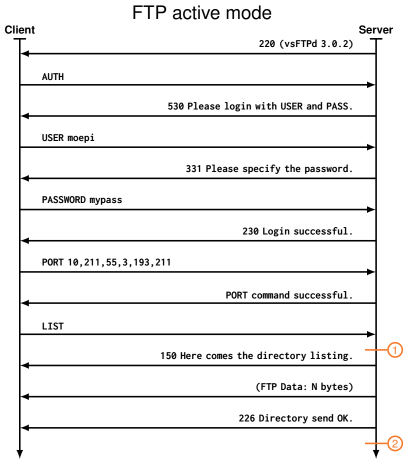


```
220 (vsFTPd 3.0.2)
AUTH
530 Please login with USER and PASS.
USER moepi
331 Please specify the password.
PASSWORD mypass
230 Login successful.
PORT 10,211,55,3,193,211
PORT command successful.
LIST
150 Here comes the directory listing.
(FTP Data: N bytes)
226 Directory send OK.
```

FTP passive mode

File Transfer Protocol (FTP)

Example:



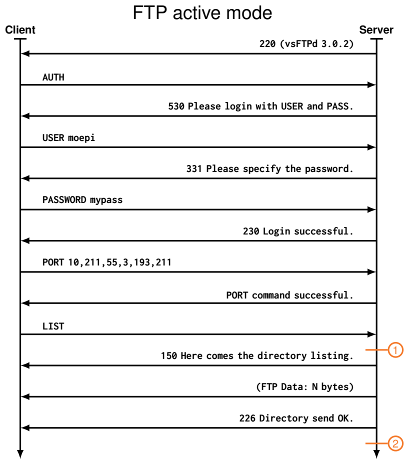
(1) Server establishes the data channel to `10.211.55.3:49619`

(2) Termination of the data channel

FTP passive mode

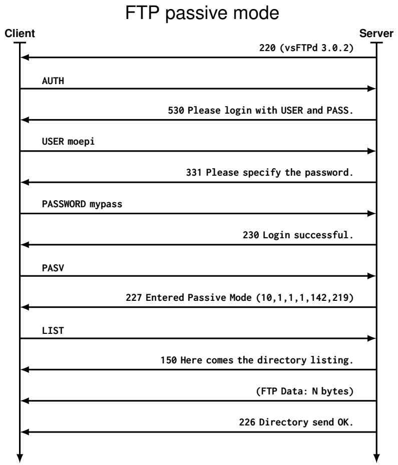
File Transfer Protocol (FTP)

Example:



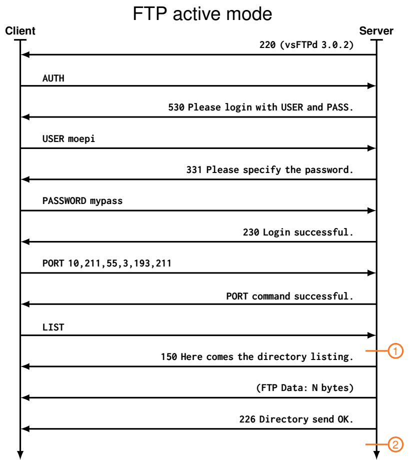
(1) Server establishes the data channel to `10.211.55.3:49619`

(2) Termination of the data channel


FTP passive mode
Client
Server
220 (vsFTPd 3.0.2)
AUTH
530 Please login with USER and PASS.
USER moepi
331 Please specify the password.
PASSWORD mypass
230 Login successful.
PASV
227 Entered Passive Mode (10,1,1,1,142,219)
LIST
150 Here comes the directory listing.
(FTP Data: N bytes)
226 Directory send OK.

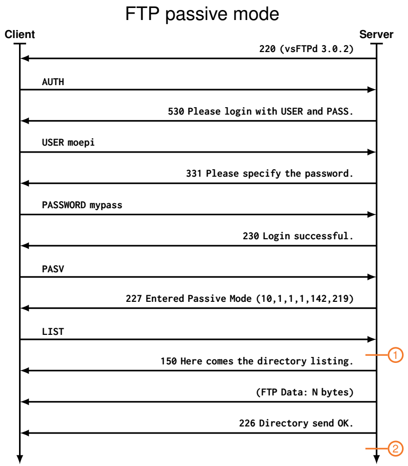
File Transfer Protocol (FTP)

Example:



(1) Server establishes the data channel to `10.211.55.3:49619`

(2) Termination of the data channel



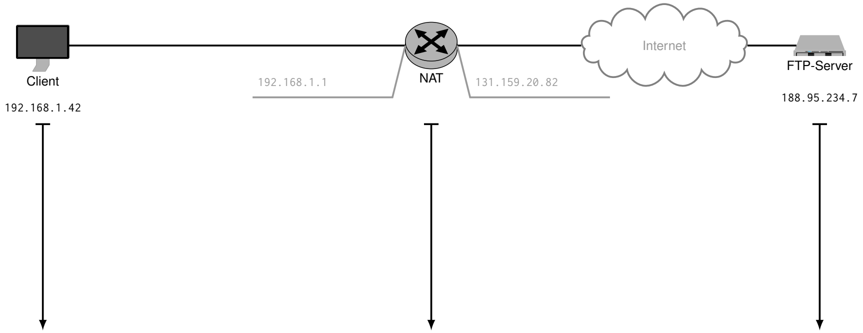
FTP passive mode
Client
Server
220 (vsFTPd 3.0.2)
AUTH
530 Please login with USER and PASS.
USER moepi
331 Please specify the password.
PASSWORD mypass
230 Login successful.
PASV
227 Entered Passive Mode (10,1,1,1,142,219)
LIST
150 Here comes the directory listing.
(FTP Data: N bytes)
226 Directory send OK.
1
2

(1) Client establishes the data channel to `<server>:36571`

(2) Termination of the data channel

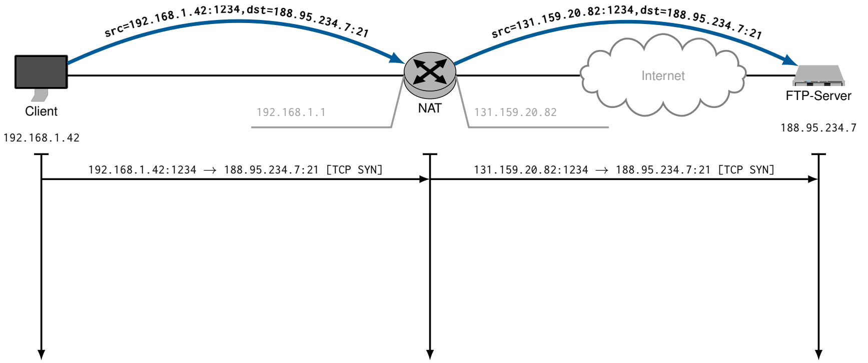
File Transfer Protocol (FTP)

FTP and NAT: What is the problem with NAT when FTP is used in active mode?



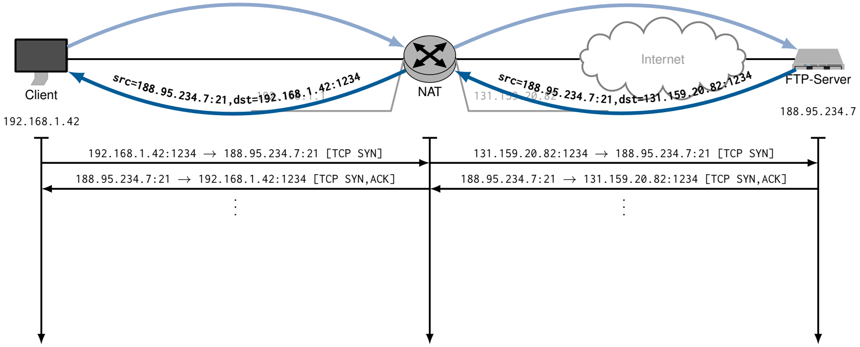
File Transfer Protocol (FTP)

FTP and NAT: What is the problem with NAT when FTP is used in active mode?


src=192.168.1.42:1234,dst=188.95.234.7:21
src=131.159.20.82:1234,dst=188.95.234.7:21
Internet
Client
NAT
FTP-Server
192.168.1.1
131.159.20.82
192.168.1.42
188.95.234.7
192.168.1.42:1234 → 188.95.234.7:21 [TCP SYN]
131.159.20.82:1234 → 188.95.234.7:21 [TCP SYN]

File Transfer Protocol (FTP)

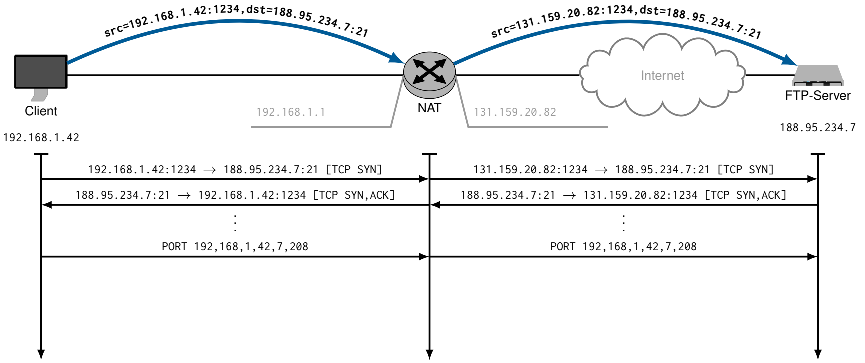
FTP and NAT: What is the problem with NAT when FTP is used in active mode?



Client
192.168.1.42
NAT
Internet
FTP-Server
188.95.234.7
src=188.95.234.7:21,dst=192.168.1.42:1234
192.168.1.1
src=188.95.234.7:21,dst=131.159.20.82:1234
131.159.20.82
192.168.1.42:1234 → 188.95.234.7:21 [TCP SYN]
188.95.234.7:21 → 192.168.1.42:1234 [TCP SYN,ACK]
131.159.20.82:1234 → 188.95.234.7:21 [TCP SYN]
188.95.234.7:21 → 131.159.20.82:1234 [TCP SYN,ACK]

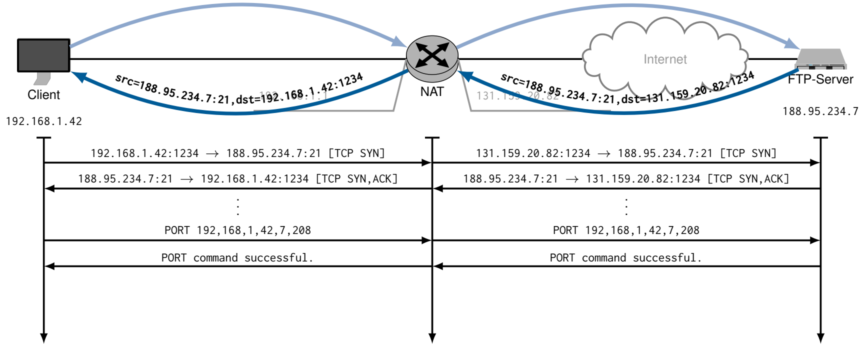
File Transfer Protocol (FTP)

FTP and NAT: What is the problem with NAT when FTP is used in active mode?


src=192.168.1.42:1234,dst=188.95.234.7:21
src=131.159.20.82:1234,dst=188.95.234.7:21
Internet
Client
192.168.1.42
192.168.1.1
NAT
131.159.20.82
FTP-Server
188.95.234.7
192.168.1.42:1234 → 188.95.234.7:21 [TCP SYN]
131.159.20.82:1234 → 188.95.234.7:21 [TCP SYN]
188.95.234.7:21 → 192.168.1.42:1234 [TCP SYN,ACK]
188.95.234.7:21 → 131.159.20.82:1234 [TCP SYN,ACK]
PORT 192,168,1,42,7,208
PORT 192,168,1,42,7,208

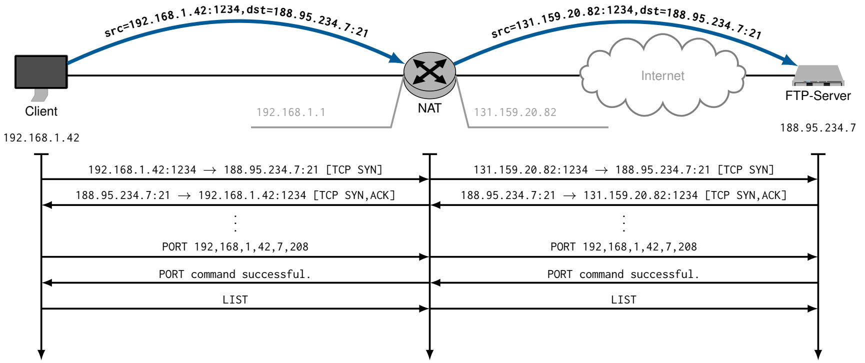
File Transfer Protocol (FTP)

FTP and NAT: What is the problem with NAT when FTP is used in active mode?


Client
192.168.1.42
NAT
Internet
FTP-Server
188.95.234.7
src=188.95.234.7:21,dst=192.168.1.42:1234
192.168.1.1
131.159.20.82
src=188.95.234.7:21,dst=131.159.20.82:1234
192.168.1.42:1234 → 188.95.234.7:21 [TCP SYN]
188.95.234.7:21 → 192.168.1.42:1234 [TCP SYN,ACK]
PORT 192,168,1,42,7,208
PORT command successful.
131.159.20.82:1234 → 188.95.234.7:21 [TCP SYN]
188.95.234.7:21 → 131.159.20.82:1234 [TCP SYN,ACK]
PORT 192,168,1,42,7,208
PORT command successful.

File Transfer Protocol (FTP)

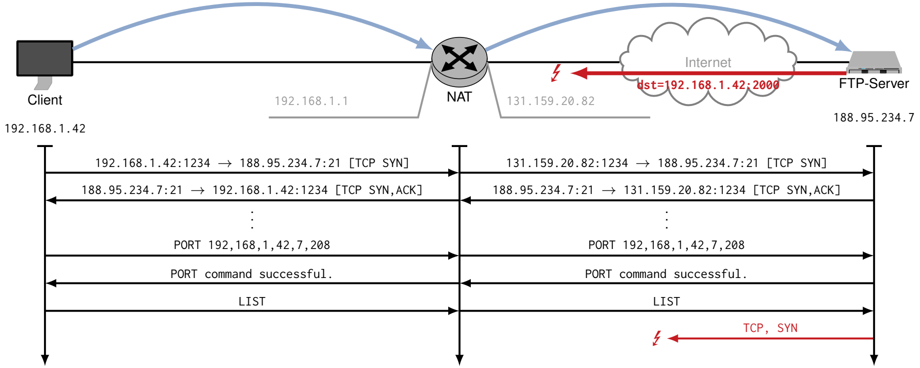
FTP and NAT: What is the problem with NAT when FTP is used in active mode?



src=192.168.1.42:1234,dst=188.95.234.7:21
src=131.159.20.82:1234,dst=188.95.234.7:21
Client
192.168.1.42
192.168.1.1
NAT
131.159.20.82
Internet
FTP-Server
188.95.234.7
192.168.1.42:1234 → 188.95.234.7:21 [TCP SYN]
188.95.234.7:21 → 192.168.1.42:1234 [TCP SYN,ACK]
PORT 192,168,1,42,7,208
PORT command successful.
LIST
131.159.20.82:1234 → 188.95.234.7:21 [TCP SYN]
188.95.234.7:21 → 131.159.20.82:1234 [TCP SYN,ACK]
PORT 192,168,1,42,7,208
PORT command successful.
LIST

File Transfer Protocol (FTP)

FTP and NAT: What is the problem with NAT when FTP is used in active mode?



- The server tries to establish a connection to `192.168.1.42:2000` as asked by the client via the `PORT` command in response to the `LIST` command.
- This will already fail due to the private IP address of the client.
- However, even if the address would be public, the NAT would not have an entry for the incoming TCP connection on port 2000.

File Transfer Protocol (FTP)

Solution:

1. The functionality of NAT must be extended such that FTP is supported:
    - the NAT must detect the FTP command channel and check the L7 PDU for the `PORT` command,
    - the contained private IP address must be substituted (the packet is modified!), and
    - a dynamic port forwarding for the expected data channel must be created.
2. FTP passive mode
    - Since the server does not establish a connection to the client in this case, but the client initiates the second channel, there should be no issue with NAT.
    - However, if the server itself is located behind a NAT that only forwards TCP 21 to a specific private address (the server), the data channel will fail again.
    - Another problem arises from firewalls that allow only connections on specific ports.

Chapter 5: Session, Presentation, and Application Layer

References

[1] R. Braden.
Requirements for Internet Hosts – Communication Layers, 1989.
https://tools.ietf.org/html/rfc1122.

[2] R. Elz and R. Bush.
Clarifications to the DNS Specifications, 1997.
https://tools.ietf.org/html/rfc2181.

[3] P. Moackapetris.
Domain Names – Concepts and Facilities, 1987.
https://tools.ietf.org/html/rfc1034.

[4] P. Moackapetris.
Domain Names – Implementation and Specification, 1987.
https://tools.ietf.org/html/rfc1035.

[5] J. Postel and J. Reynolds.
File Transfer Protocol (FTP), 1985.
https://tools.ietf.org/html/rfc959.