8 Application Layer

8.1 Application layer in the ISO/OSI protocol stack

The ISO/OSI protocol stack distinguishes three application layers with different tasks:

- Session layer, link layer, communication control layer (layer 5),
- Presentation layer (layer 6),
- Application layer (layer 7).

8.1.1 Session layer

The **session layer (Layer 5, Link Layer, Communication Control Layer)** is responsible for coordination and communication between different application processes. It enables the establishment of shared sessions among users on different communication devices. It completes connections between processes and applications on different host computers.

Special tasks are

- Process synchronization,
- Establishing process-to-process connections,
- User authorization management (names, passwords),
- Logon of a user to a session on a remote system,
- Transfer of files between sender and receiver,
- Dialog control between session participants,
- Token management in case two session participants are not allowed to perform the same operations,
- Conversion of transmitted data packets into “application data",
- Ensuring data security.

8.1.2 Presentation layer

The **presentation layer (layer 6)** provides the application-specific coding, formatting and interpretation of data structures and data formats. It provides general solutions for frequently used functions (e.g. printer, monitor, file functions). In contrast to the other layers, it also takes care of the syntax and semantics of information transfer in addition to error-free data transfer.
The presentation layer defines the form of message presentation to the user (text, graphics, image, film, audio, etc.). Hardware differences between different network components are covered. It is often implemented in the form of callable routines and libraries.

The special tasks of the presentation layer are

- Abstract definition of the data structures to be transmitted,
- Bidirectional conversion between the internal representation of the communication terminals and the standard representation of the network,
- Data compression.

8.1.3 Application layer

The **application layer (layer 7, processing layer)** provides a wide variety of protocols for different applications, e.g. protocols for different terminal types, graphics applications, control sequences, etc. It manages information about the applications and programs, as far as these are important for the operation in the network. It also takes care of network management.

Special tasks of the processing layer are thus

- Provision of an abstract virtual network terminal for the adaptation of editors and programs,
- Mapping of the virtual network terminal to the physically existing terminal,
- File transfer between different systems and provision of the necessary conversion facilities, e.g. for e-mail, data transfer between different operating systems, etc.,
- Provision of programs for network users,
- Network management.

At the application layer level, the coupling of different subnetworks is possible by using **gateways**. In contrast to routers, gateways enable the coupling of different subnetworks with completely different addressing techniques and incompatible protocols.

Because of the enormous software effort required to implement a gateway, the delay times for data transmission are relatively long compared to routers.

(In TCP/IP networks, routers are sometimes (incorrectly) also referred to as gateways).

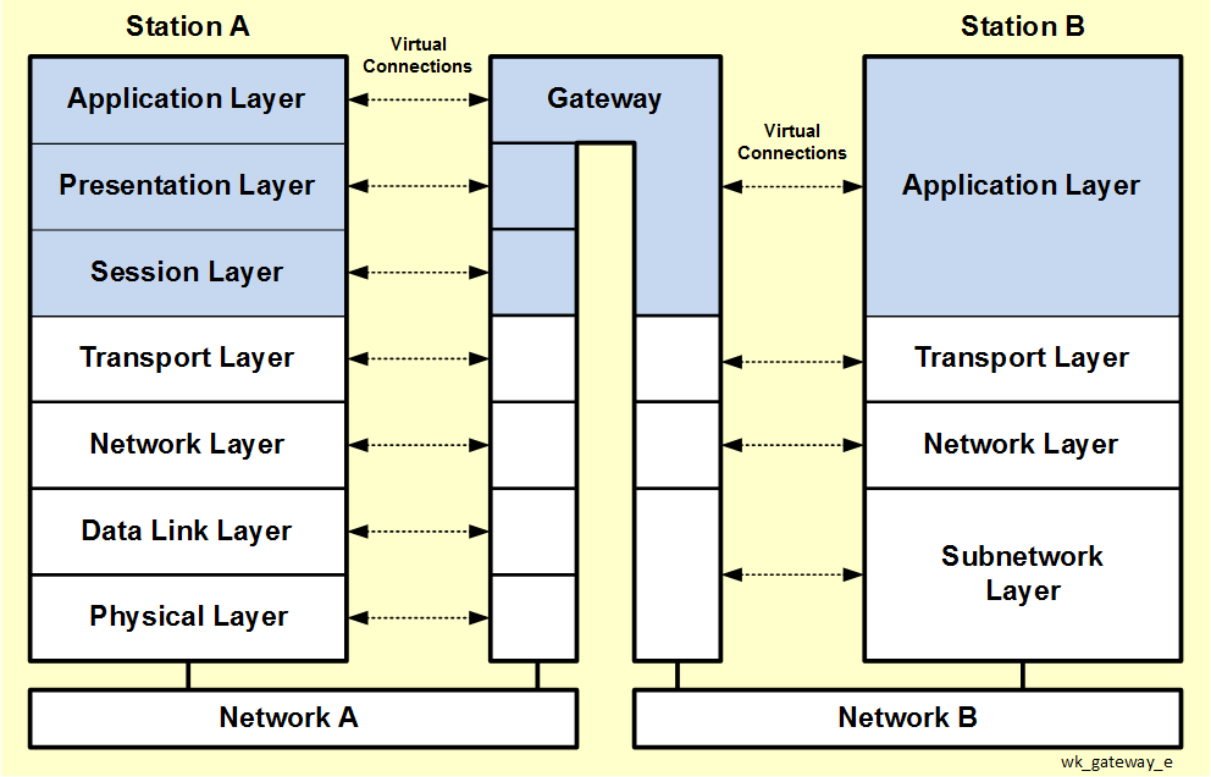


Fig. 8.1.3-1: Schematic representation of a gateway between two networks

8.2 Application services using TCP/IP

TCP/IP combines the three different application layers of the ISO/OSI model in one layer, the application layer.

Services and protocols of the TCP/IP application layer can be divided into two categories:

- **Services and protocols for the end user:**

  They are used directly by the end user.

  Examples:

  - Communication on the Internet using HTTP (Hyper Text Transfer Protocol),
  - Interactive login with Telnet,
  - Secure interactive login with SSH (Secure Shell),
  - File transfer with FTP (File Transfer Protocol),
  - Email dispatch with SMTP (Simple Mail Transfer Protocol).

- **Support services and protocols:**

  They are required to support the services and protocols for the end user in the background.

  Examples:

  - Name mapping with DNS (Domain Name Service),
  - Ensuring secure authorization, user and data integrity, confidentiality with TLS (Transport Layer Security),
  - Inter-process communication RPC (Remote Procedure Control),
  - Network management with SNMP (Simple Network Management Protocol).

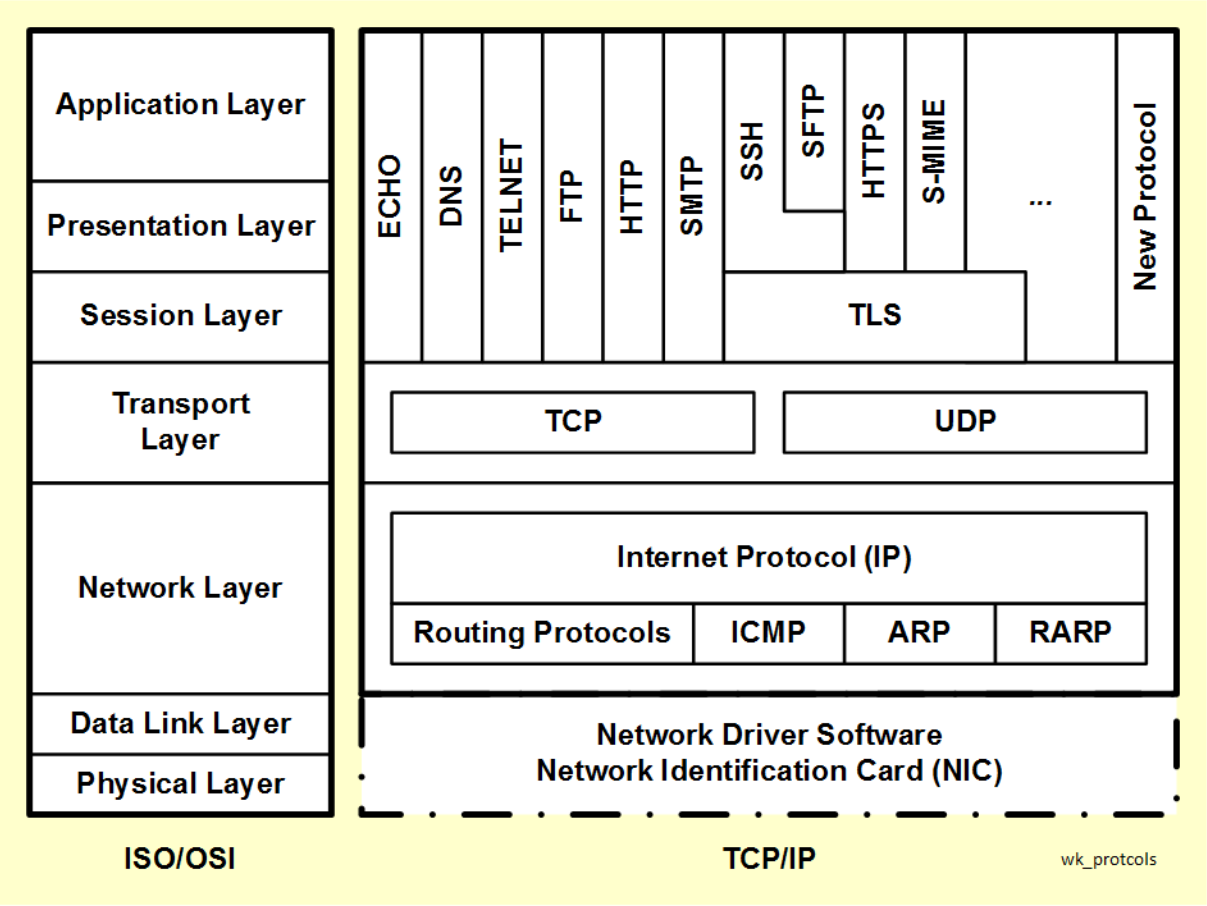


Fig. 8.2-1: Some protocols in the TCP/IP protocol stack

8.3 Address management with the Domain Name Service DNS

8.3.1 Domains and domain names

The Domain Name System DNS is a dynamic, decentrally organized directory service for managing and resolving symbolic names in communication networks. Decentralized name servers manage a fixed network area (domain). There is an overlapping communication for the data traffic between the domains.

Each domain determines itself how the domains created under it are managed.

The address hierarchy within a domain usually has a hierarchical tree-like structure. Organizationally related network parts, e.g. corporate networks, are grouped together in non-overlapping DNS zones.

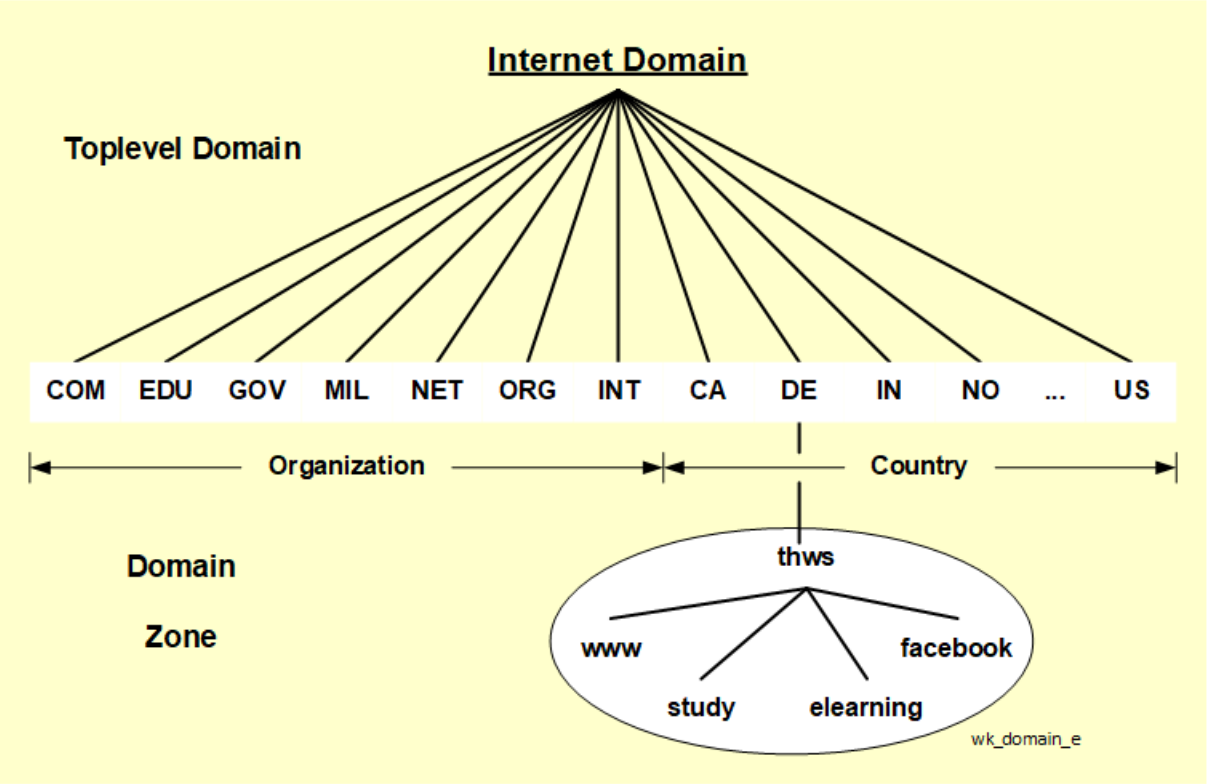


Fig. 8.3.1-1: Hierarchical organization of the Internet domain structure

Each zone generally operates a name server (primary server) and, for security reasons, a redundant replacement server (secondary server).

The lowest hierarchy level in the name tree is formed by the "toplevel" domains. The toplevel domains were introduced in the pioneering years of the Internet and distinguish different organizational forms in a general area (generic area). Later, a country code was added:

Domain name	Description
COM	Commercial organization
EDU	Educational institutions in the US
GOV	Government institutions in the US
MIL	Military organizations in the US
NET	Network service facilities
ORG	Other organizations
INT	International organizations
ARPA	ARPANET domain
country code	Country code

Examples of country codes (not complete):

Abbreviation	Country	Abbreviation	Country
AU	Australia	*FR*	France
BE	Belgium	*IN*	India
CA	Canada	*IT*	Italy
CH	Switzerland	*JP*	Japan
CN	China	*NL*	Netherlands
DE	Germany	*NO*	Norway
ES	Spain	*US*	USA

In Germany, the cooperative DENIC eG (German Network Information Center), a governmental organization, centrally administers the top-level domain .DE.

The address notation is carried out by introducing domain names. The simple names are arranged one after the other, separated by a decimal point. When specifying names, the hierarchy tree is traversed from the lower branches towards the unnamed root, i.e. the closest domain is named first, the furthest domain last.

Absolute name specifications are terminated with a dot. Relative name specifications do not have a terminating dot.

Examples:

```
study.thws.de
physik.uni-wuerzburg.de
freseniusmedicalcare.com
```

There is no distinction between upper and lower case. Names can be a maximum of 63 characters long. The entire path specification may contain a maximum of 255 ASCII characters.

8.3.2 Name resolution

Name resolution (mapping) is performed on an application-oriented protocol layer by converting the symbolic name into the Internet address.

For name resolution, a client/server connection is established between the resolver client and the name server.
The client sends a request (query) to the name server specifying the logical name to be resolved and the desired type of resolution:

- **nonrecursive resolution (iterative resolution):**

  - The name server performs the resolution itself, or, if it cannot perform the mapping itself, determines another name server with more and informs the client of its address in a response.
  - The client starts a new request, etc.

- **recursive resolution:**

  - The name server is responsible for the complete mapping.
  - The resolved name is returned recursively to the calling client (relief of the clients from request activities).

The data transport of the requests and responses can, depending on the implementation, take place via the transport protocols UDP and TCP.

The determination of the symbolic name with a known IP address is done by the inverse name resolution (inverse mapping).
A special domain is set up for this purpose: IN-ADDR.ARPA.

The corresponding information is taken from a database and made available to the requesting party. The IP addresses are stored in inverse notation and must also be queried in this form.

Example:

Request for the symbolic name of the address **`192.134.75.3`**:

```
3.75.134.192 IN-ADDR.ARPA
```

8.3.3 Resource record

Each host in a data network is identified not only by its address (IP address), but also by a five-digit data record, the resource record.

The Domain Name Service implements a mapping from domain names to resource records. The five components of each resource record are stored line by line in the database of the responsible DNS server:

```
Domain_name    Time_to_Live   Type      Class     Value
```

Explanations:

- **Domain Name:**
  - Name of the domain to which the record refers,
  - Primary search key for database queries,

- **Time to Live:**
  - Stability of the record over time (specified in seconds),

- **Type:**
  - Type of resource record,

  Important types:

Type	Meaning	Value
SOA	Start of Authority	parameter for the zone in question
A	IP address of a host	32-bit integer
MX	Mail Exchange	Priority for e-mail
NS	Name Server	Name of the domain name server
CNAME	Canonical name	Domain name
PTR	Pointer	Alias name for an IP address
HINFO	Host description	CPU and operating system (ASCII)
TXT	Text	uninterpreted ASCII text,

- ***Class:***
  - Class for the information, e.g. IN for Internet information,

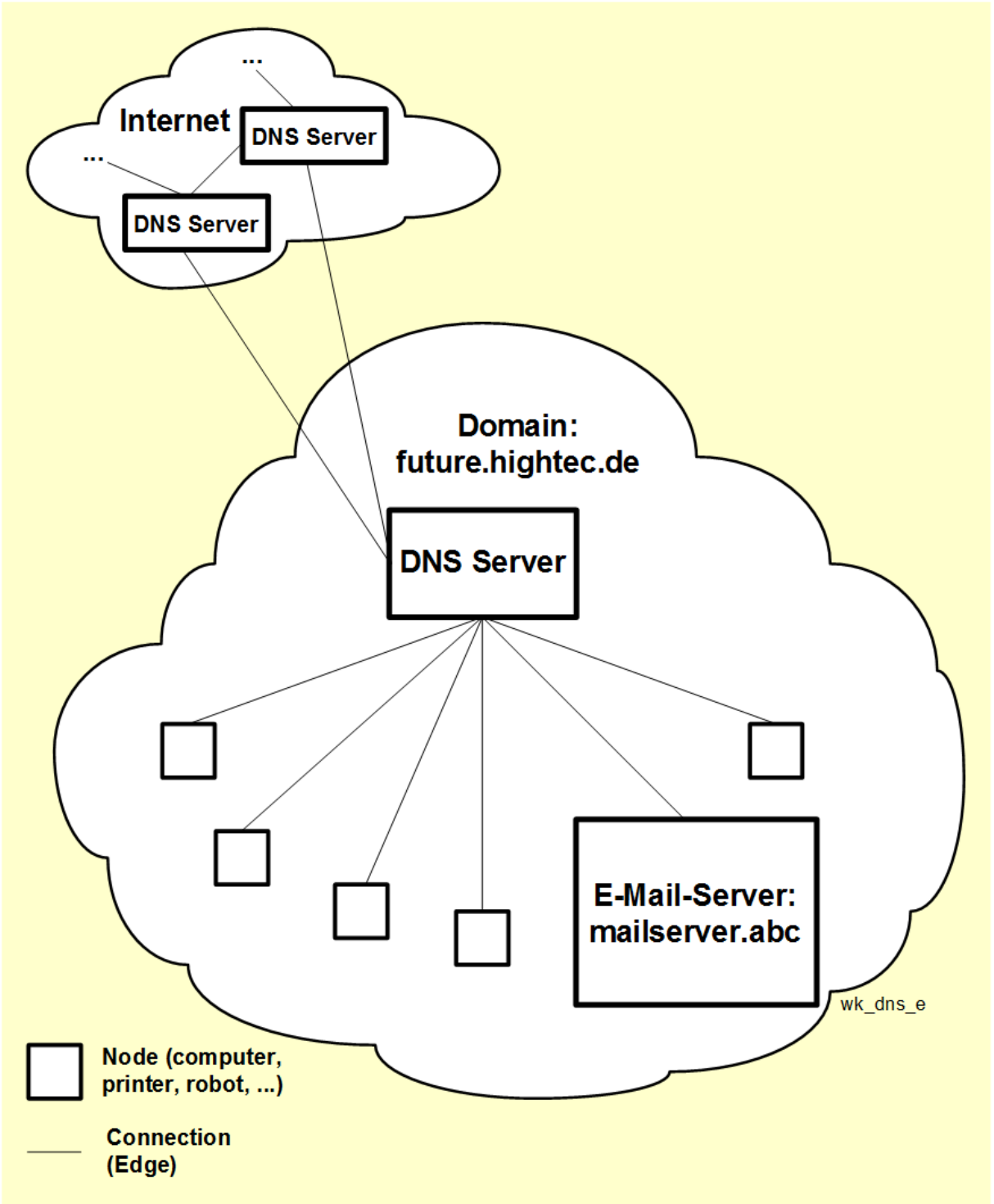
- ***Value:***
  - Number, domain name or ASCII string,
  - Semantics depending on record type (type).

Each domain can have multiple resource records associated with it. The resolver passes the domain names to the DNS server.

Exercise

E.8.3.3-1: In a data network with the domain **`future.hightec.de`** all incoming mails are to be forwarded to the email server **`mailserver.abc`**.

Formulate an entry for the responsible DNS server.



8.3.4 DNS message

A **Domain Name Service system data packet (DNS message)** comprises several sections:

Header, Question, Answer, Authority, Additional Information Sections.

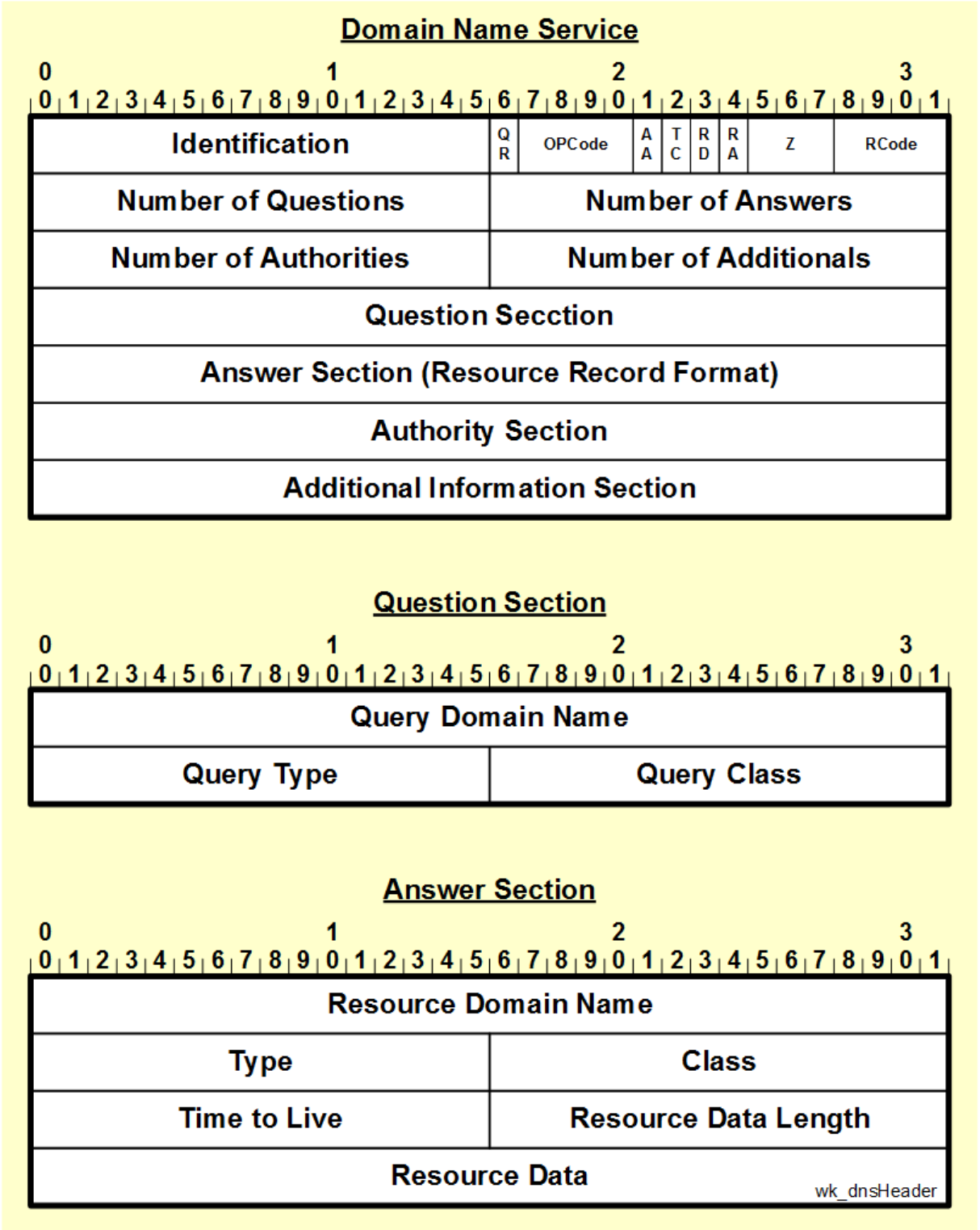


Fig. 8.3.4-1: Header of the DNS service

The following list explains the various subsets:

- **Identification:**
  Identification number for correct assignment of request and response,

- **Parameter:**
  Specification of the type of request and response:

Bit	Parameter	Meaning
0	QR	0 Request (query) 1 Response
1 ... 4	OPCode	option code type for defining the type of query (will be set in query and copied in response): 0 Standard 1 Inverse 2 Server status 3 ... 15 Reserved
5	AA	Authorative Answer: set if the domain server is an authority (only for DNS responses),
6	TC	Truncation: set, if message is invalid (maximum allowed length exceeded),
7	RD	Recursion desired: set, if from Subdomain Name Server the Internet address for a logical name is expected,
8	RA	Recursion available: set, if name server supports the recursion mechanism (only for replies),
9 ... 11	Z	Z-Field: reserved for future developments (value zero)
12 ... 15	RCode	Response Code (only for DNS response): 0 No error 1 Format error in request 2 Server error 3 non-existence of the name 5 Rejection of the response by the name server 6...15 Reserved.

- **Number of Questions:**
  Number of questions in the Question Section,

- **Number of Anwers:**
  Number of answers in the Answer Section,

- **Number of Authority:**
  Number of resource records in the Authority Section,

- **Number of Additionals:**
  Number of resource records in the Additional Information Section,

- **Query Domain Name:**
  Sequence of Labels:
  - Each label consists of length octets and data.
  - Each domain name is terminated with a null label.

- **Query Type:**
  Type of query,

  Examples (not complete):

Name	Value	Description
A	1	Host address
NS	2	Authority name server
CNAME	5	Alias name
TXT	16	Text string
...		

- **Query Class:**
  Classification of requests into classes:

Class	Value	Description
IN	1	Internet
CS	2	No longer in use
CH	3	Chaos class
HS	4	Hesiod
*	255	all classes

- **Answer Section:**
  Identical to Resource Record Section,

- **Resource Domain Name:**
  Domain name for which the information is stored in the Data field,

- **Type:**
  Data type in the data field (types as in the Query Type field),

- **Class:**
  Class division in the Data field (like Query Class),

- **Time to Live:**
  - Maximum storage time (in seconds) of a resource set message in the cache memory of the receiver,
  - After expiration of the available time, message is deleted,

- **Resource Data Length:**
  Length (in bytes) of the data field,

- **Resource Data:**
  Storage of the actual data,

- **Authority Section:**
  Identifies the server that supplied the actual information, and the information,

- **Additional Information Section:**
  - Additional information, e.g. about servers for e-mail exchange,
  - Entries according to the Resource Record Format.

8.4 HTTP Protocol

8.4.1 Functionality

The **Hypertext Transfer Protocol HTTP** (RFC 2616, extension: RFC 2817) is the basis for transferring information of various types (normal text, hypertext, audio data, images, any Internet-compatible data) in the World Wide Web (WWW), the framework for accessing documents stored on computers distributed on the Internet.

The protocol is used in client/server applications. It works as a "stateless" protocol: each transaction, i.e. each transfer of a partial object, e.g. a partial image, is independent of the previous transaction. The TCP protocol (TCP port 80) is used for data transmission. After completion of each transaction, the virtual connection between client and server is terminated again.

The communication between client and server works according to the following scheme:

- The client browser (user agent) initializes a TCP connection to the server that provides the web page.
- After the connection has been successfully established, the client sends a HTTP request to the server, consisting of a special command (method), the unique identifier of the data object (Uniform Resource Locator URL) and, if necessary, further additional information (parameters, information about the client, ...). The request is coded according to the ASCII code.
- After receiving the client request, the server processes the request and sends the result back to the client, including status information, confirmation message and possibly necessary additional information (HTTP response). The response has a MIME-like format.
- After receiving the response from the server, the connection is closed.

8.4.2 Unique identification of documents on the Internet

Documents stored on the Internet are uniquely identified by a **Uniform Resource Locator URL**. A Uniform Resource Locator consists of three parts, which are composed in the following form:

`AccessMethod://Computername[:Port]/DocumentIdentifier#Anchor`.

The following rules must be obeyed in the formulation:

- The access method specification is separated from the other specifications with a colon.

  Examples of access methods:

`http:`	Web access (Hypertext Transfer Protocol),
`https:`	Secure web access (Secure Hypertext transfer protocol),
`telnet:`	Access and working on a remote computer,
`ssh:`	Secure access and working on a remote computer (SSH: Secure shell),
`ftp:`	copying a file from a remote computer,
`finger:`	addressing files on the hard disk of the local computer.

- The (logical) name of the computer is preceded by a double slash and terminated by a single slash.
- The computer name can be replaced by specifying the Internet Protocol address (IP address).
- The specification of the port number is optional.
- The identification of the document includes the path specification and the file name of the document.
- An optional specification of an anchor point, a marker in a HTML file, is possible. An introductory double cross serves as marking.

Examples:

https://www.thws.de

https://studium-robotik.thws.de/en/

https://studium-robotik.thws.de/wp-content/uploads/2023/03/Modulhandbuch-Robotik-IRO Dez2022_EN_THWS.pdf

8.4.3 Intermediate systems

If a direct connection between the client (user agent) and the server is to be avoided, it is possible to install intermediary systems in the form of **gateways** and **proxies**.

Gateways and **proxies (proxy servers)** act from the client's point of view as if they were the actual servers. They are used when the client does not have direct access to the target server. Applications are often found in secure data networks where the individual clients do not have direct access to the open Internet (firewall systems).

Another application focus is the implementation of different access methods. For example, if the FTP protocol is not supported by a browser (client), an intermediate proxy can take over the translation between HTTP protocol and FTP protocol.

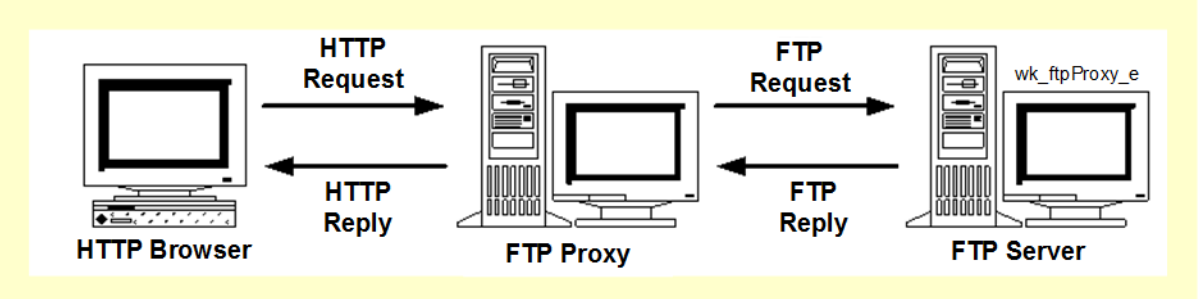


Fig. 8.4.3-1: Proxy application: Scheme of FTP connection with HTTP browser that does not support FTP protocol

Example:

Downloading the `sphero.twist` software package from GitHub:

https://github.com/Schweinfurt/sphero_twist

The call for downloading the package using the FTP protocol (FTP: File Transfer Protocol) remains hidden.

8.5 Logging in and working on a remote computer with Telnet

8.5.1 Telnet functionality

The **Telnet protocol** (RFC 854) allows a user to log on to a remote station in the communication network **(Standard Remote Login Protocol)**, edit files, start applications and control peripheral devices, e.g. printers. The protocol establishes a client-server connection between the **local client software (client)** and the remote **server software (server, Telnet daemon)** and handles data transfer between the stations. Although only the local user interface is used, when the connection is open, it appears to the user as if he or she is logged on directly to the remote host.

The Telnet protocol performs three tasks:

- Definition of a **virtual network terminal (Network Virtual Terminal NVT)** as an interface to the remote station, independent of the respective computer environment,

- Negotiation of settings between local Telnet client and remote Telnet server for the current Telnet session,

- Symmetrical view of both sides during a session:
  - Telnet connections can be used bidirectionally by two or more users.
  - All settings proposed by one station must be confirmed by the remote station with an acknowledgement message.

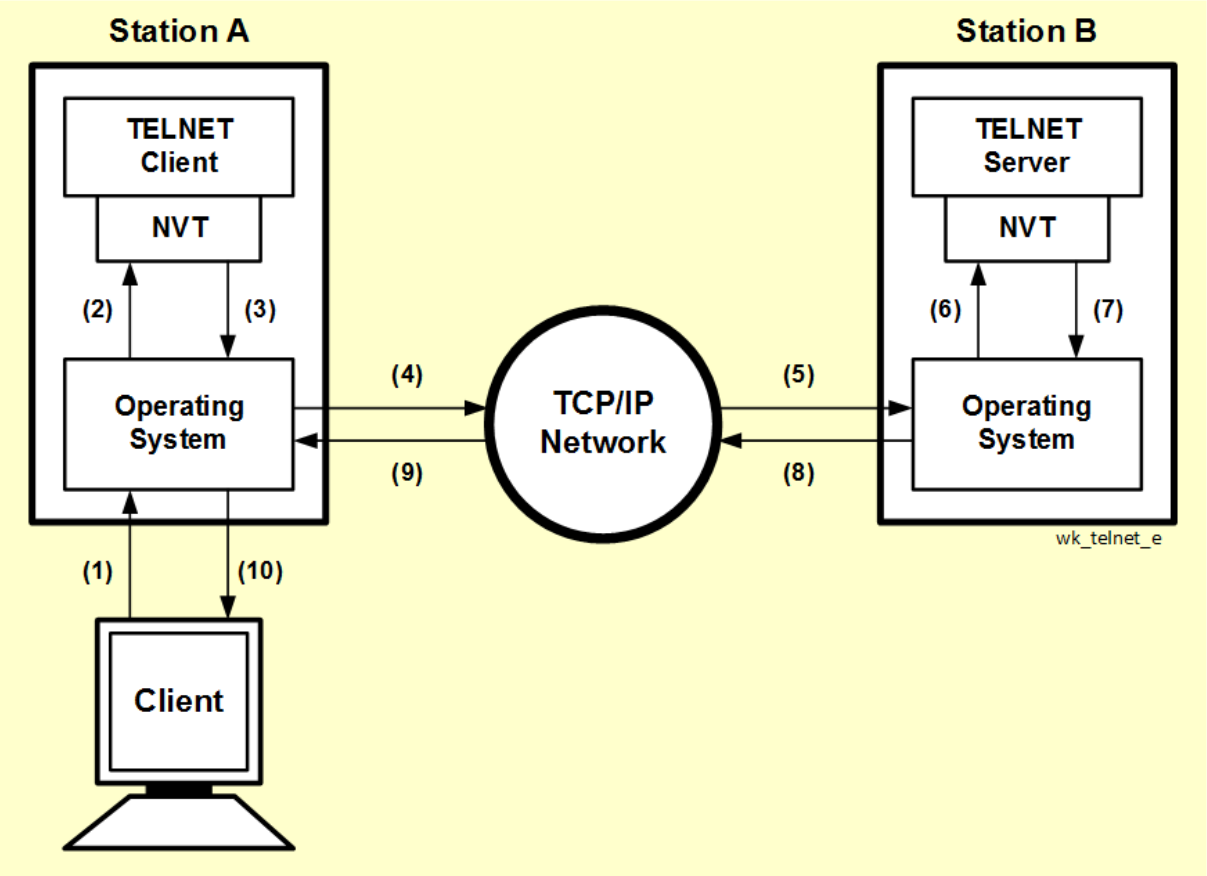


Fig. 8.5.1-1: Schematic graphical representation of a Telnet connection

The Telnet server directs the dialog between two communication partners and provides the services for the virtual network terminal. The client receives the user's commands at an interface and transforms them into commands for the virtual network terminal. On the other hand, the client receives the server's responses and transforms them into a form that the user can understand. Telnet connections are implemented using the TCP transport protocol with port 23. Telnet servers can handle requests from multiple clients.

Telnet software uses two modes of operation:

- **Command mode** with commands and subcommands for control
- (definition of terminal type, character or line by line transmission, etc.),

- **Input mode** for communication.

Example of a Telnet login:

```
$ telnet farhost
Trying 195.28.12.3
Connected to supercell.brain.com.
Escape character is '^]'.

$ login: socrates
$ password:

Kill is Ctrl-U
Interrupt is Ctrl-C
$
```

After successfully completing the login procedure, all operating system commands (in accordance to the installed security rules) that are possible on the remote station can also be used on the local station.

8.5.2 Secure login at a remote Data station

The original Telnet protocol transmits all data, including the password, in plain text (ASCI code).

The **SSH protocol (SSH: Secure SHell)** was developed for secure, encrypted data transmission when logging in to a remote data station. It enables user authentication, grants user and data integrity as well as confidentiality.

8.6 File transfer with the File Transfer Protocol FTP

8.6.1 Introduction

The **File Transfer Protocol FTP** (RFC 959, extensions: RFC 2228, 2640) enables the transfer of files via a TCP connection between two stations with any operating system and the storage of the data in the respective format used.

8.6.2 Phases of the data transfer

Data transmission is based on the client/server model and consists of five separate phases:

- **Phase 1: Connection setup of the control process**

  - Starting the FTP client software on the local system and establishing a control connection from client port p1 to server port 21 (1),
  - FTP server responds to the action with a confirmation message and the information about availability (2),
  - FTP client completes the three-way handshake with confirmation (3),
  - Client transmits its own user name via control connection (4),
  - Confirmation from server after reception (5),
  - Sending of password by local client station (6),
  - Confirmation of password input by the server (7),
  - Sending the transmission options, e.g. sending binary files, by the local client station (8),
  - Confirmation by the server (9).

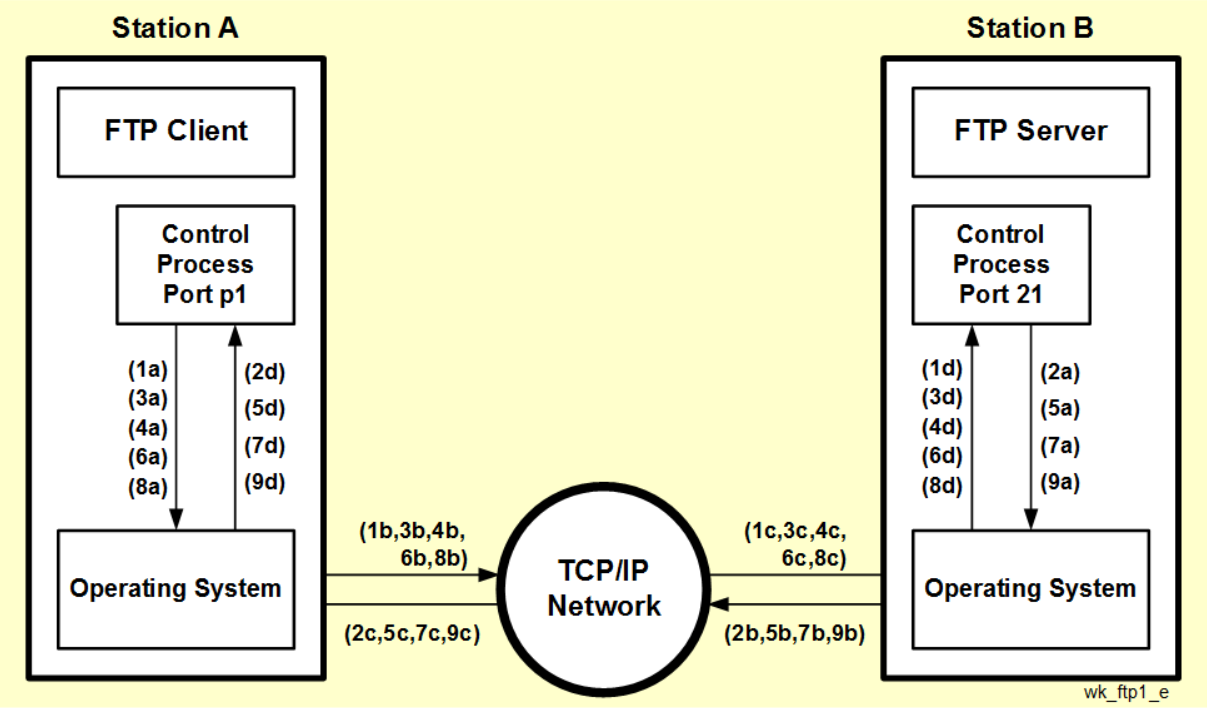


Fig. 8.6.2-1: FTP connection setup of the control process, phase 1

- **Phase 2: Generation of the data processes**

  - Local client station sends a request via the control connection for a data connection with client port p2 on the local station (10),
  - Generation of a server data process with port number 20 on the remote station by the remote FTP server, acknowledgement to the client and generation of a client data process with port number p2 on the local client station (11).

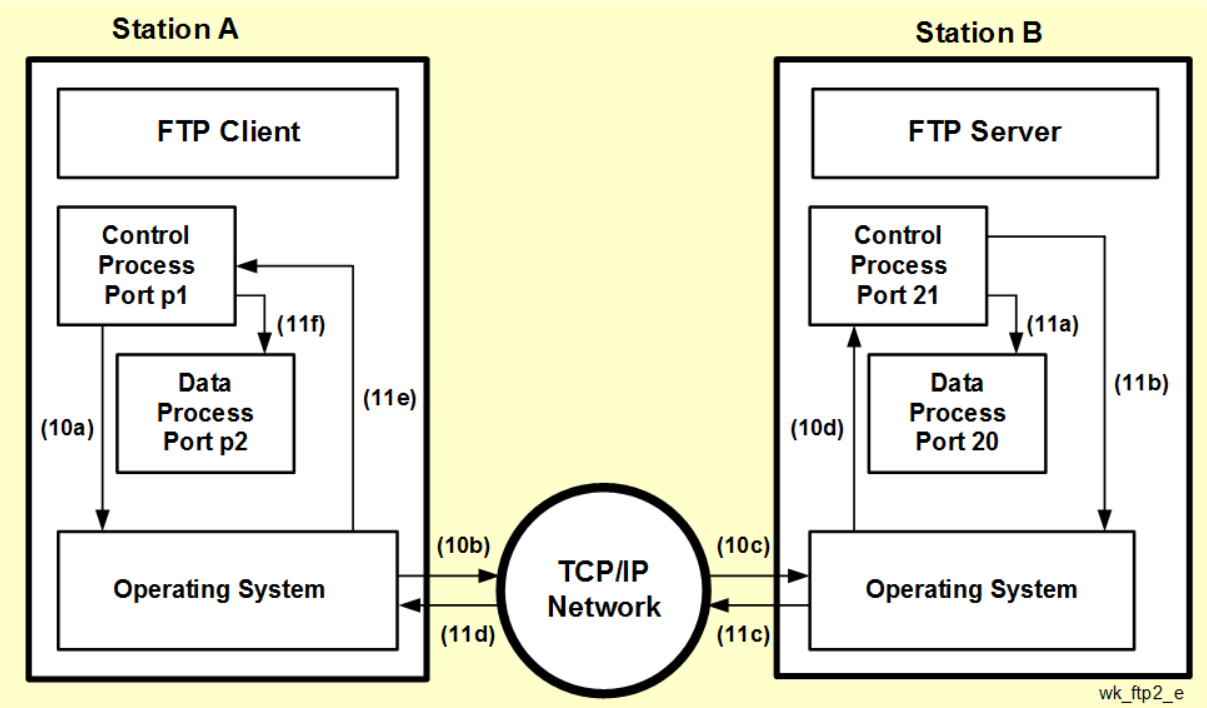


Fig. 8.6.2-2: FTP connection setup: data process generation, phase 2

- **Phase 3: Data transmission**

  - Establishing the actual TCP data connection with the destination port p2 by the remote FTP server:
    Server data process reads buffered data blocks from the file requested by the
    client and transfers them in secure segments to the client data process (port p2) over the network (12),
  - Acknowledgment of the received segments by acknowledgement messages for each individual segment without windowing procedure or in each case after the transmission of a certain number of bytes (13).

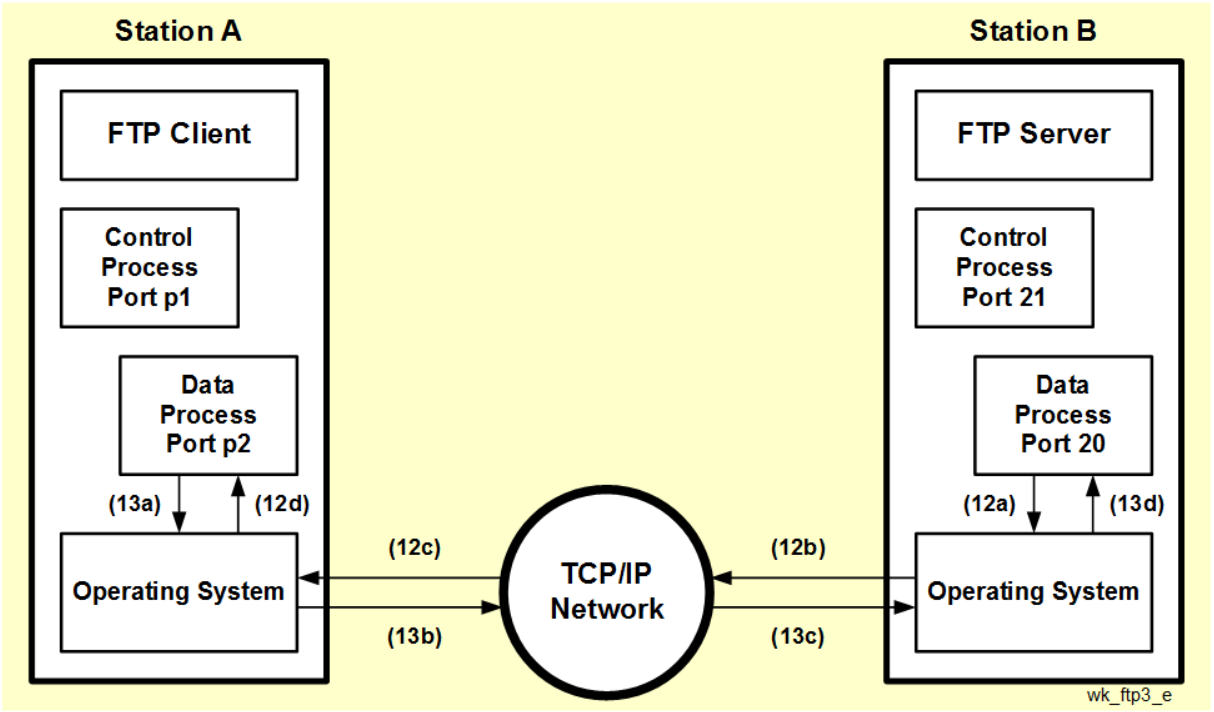


Fig. 8.6.2-3: FTP data transfer, phase 3

- **Phase 4: Initiation of end of transmission**

  - Transfer of the remaining data by the server process (14),
  - Confirmation of receiving the complete data by the client (15),
  - Initiation of the termination of the data connection by the FTP server and notification to the client with the "Close" command (16),
  - Confirmation of the connection termination by the client (17).

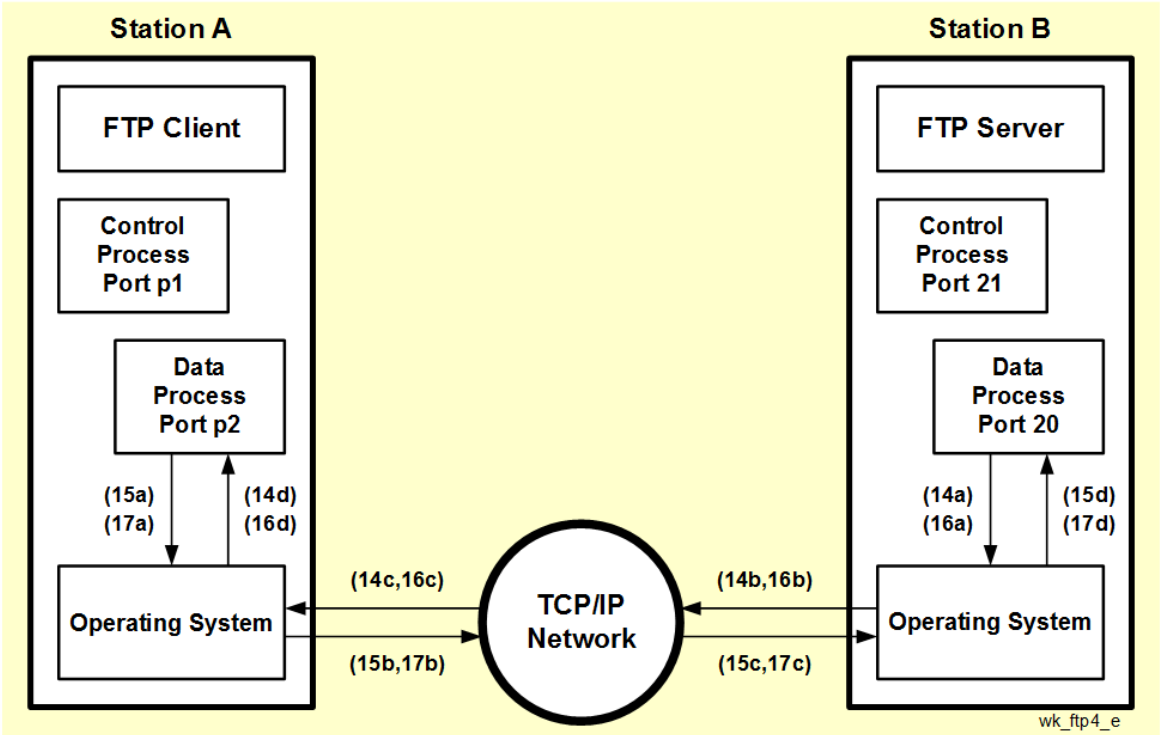


Fig. 8.6.2-4: FTP data transfer, initiation of end of transfer, phase 4

- **Phase 5: End of data transfer**

  - Server data process notifies its control process (port 21) of the end of the data transmission (18),
  - Client data process informs its control process (process p1) and terminates the data process (19),
  - Control connection is still active and can be used for new data transfers.

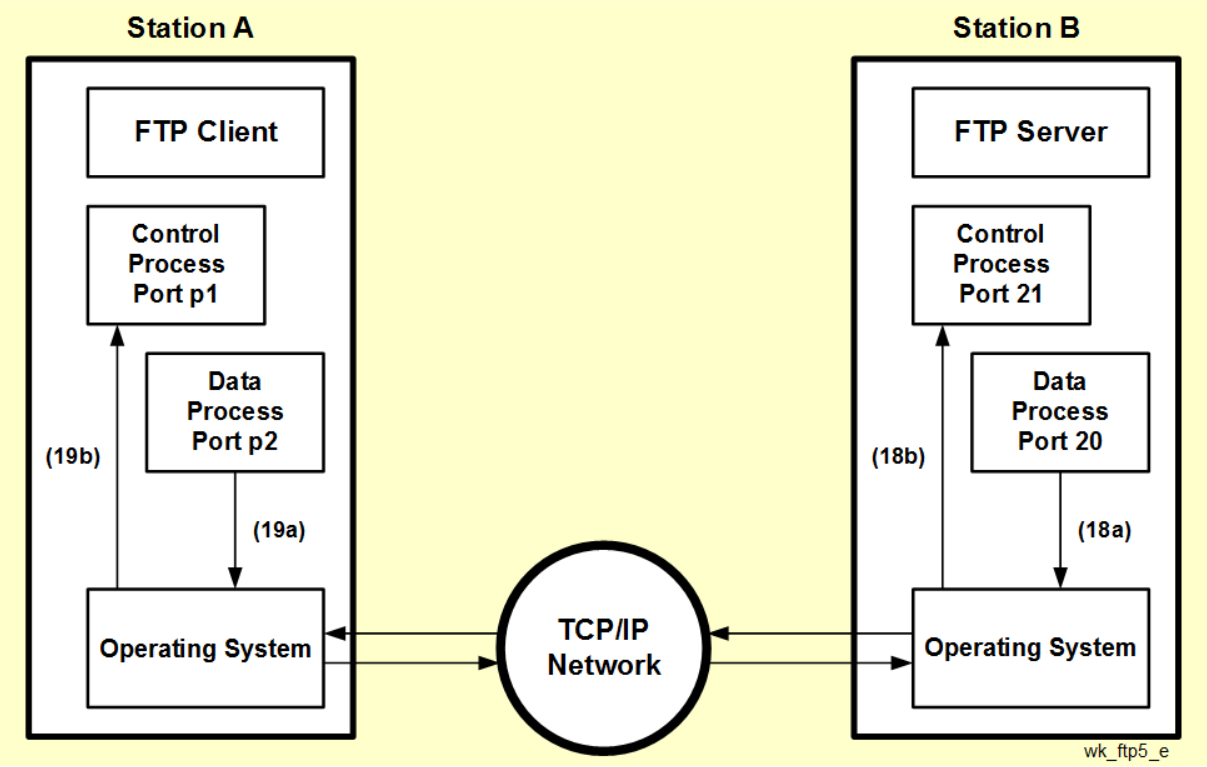


Fig. 8.6.2-5: End of FTP data transfer, phase 5

Example: Opening an FTP session and logging into the remote system

`$ftp`	Opening the FTP session
`ftp>`	Prompt character for input
`open computer2`	Select the remote system with its name (here: `computer2`) and open it for the data transfer
`connected to computer2` `FTP server ready`	Confirmation of correct opening
`Name:`	Input of user name on the remote system
`Password:`	Password on the remote system

The table below lists some FTP commands:

Command	Action
`binary`	Switching on the binary transfer mode (for binary data files, executable program code)
`cd`	Change to directory on the remote system
`del`	Delete file on the remote system
`dir`	Display directories on the remote system
`lcd`	Change directory on the local system
`get`	Copy file from remote system
`put`	Send file to remote system
`pwd`	Output current directory of remote system
`quit`	Termination of FTP session

8.6.3 Special operating modes of the FTP protocol

Active FTP

In active FTP mode, the client opens a random port to establish the FTP control connection and informs the server of this port and its own IP address.

If the client is located behind a firewall in a private subnet, the server cannot reach the client. Active mode is usually preset.

Passive FTP

In passive FTP (Passive Mode, PASV), the client automatically initializes port 21 for establishing the control connection and port 20 for data transmission.

This allows the data packets to be forwarded to the client via the NAT protocol (NAT: Network Address Translation (usually installed on a firewall)).

Passive mode is activated with the `PASV` command.

8.6.4 Implicit FTPS and explicit FTPS

For **secure file transfer**, the FTP protocol (RFC 959) has been extended by the **FTPS protocol** (RFC 2228).

Two working modes are provided:

- **Implicit FTPS:**
  The connection is made using the TLS protocol (SSL) and is always encrypted.

- **Explicit FTPS:**
  The client must explicitly request encryption from the server. If this request is missing, the server can decide for itself whether to close the connection or transmit the data unencrypted.

8.7 Electronic mail

8.7.1 Standard email

The **Simple Mail Transfer Protocol** (SMTP, RFC 821, 822; new: RFC 2821, 2822) is the standard protocol for sending and receiving electronic mail (e-mail) on the Internet. The protocol is based on a client-server architecture, which uses the TCP protocol and port address 25.

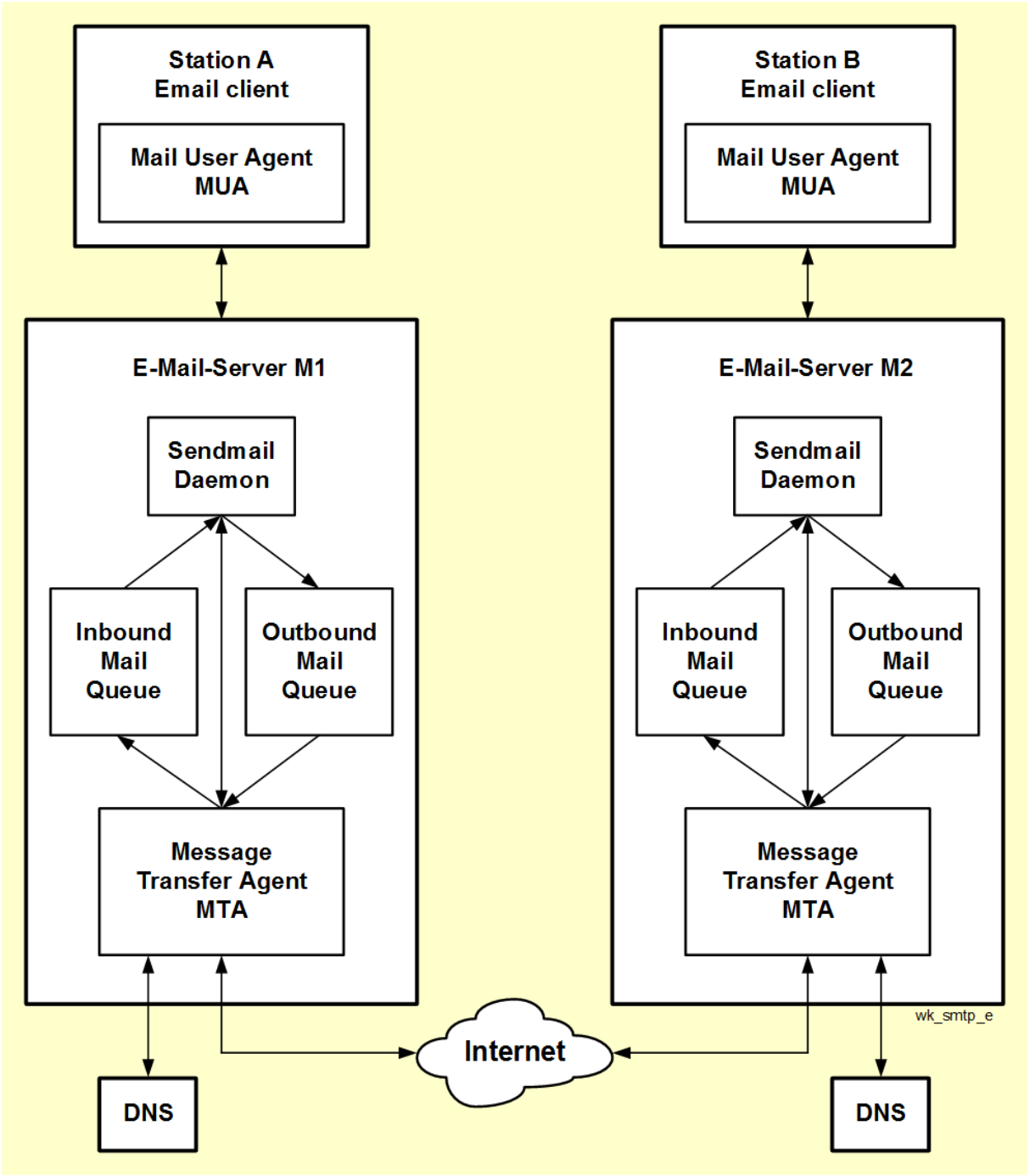


Fig. 8.7.1-1: Schematic model of SMTP connection

Transmission is based on the classic **"store-and-forward"** principle. Both the outgoing mail and the received messages are first buffered in buffer memories. All commands, parameters and messages are transmitted unencrypted in ASCII code.

The sender (client) and recipient (client) write, read and archive the mail using the **Mail User Agent (MUA)** software facility. The MUA retrieves the mail from the mail server and sends the messages to be sent to the mail server.

Historically, the mail server was a Unix machine with a running sendmail daemon. The mail server stores and buffers incoming mail for users and runs what is called the **mail transport agent (MTA)**. The MTAs provide the software platform for transport between the various servers. The underlying protocol is the **Simple Mail Transfer Protocol (SMTP)**.

Incoming mail can be read by a **Mail User Agent (MUA)** installed directly on the mail server; or it can be picked up from a remote terminal via a Mail Access Protocol, e.g. **POP3** (Post Office Protocol, Version 3, RFC 1939, new: RFC 2449) or **IMAP4** (Internet Message Access Protocol, Version 4, RFC 2060, new: RFC 3501).

The addresses of the mail servers (**`user@email_server`**) are resolved by the **Domain Name Service (DNS)**. After name resolution or conversion to the IP address of the remote machine, a TCP connection is installed and the mail can be transmitted.
Verification of the sender address is optional and implementation-dependent.

The SMTP protocol uses fixed commands from the sending device and confirmation messages from the recipient for communication. Each transmitted command is returned with an acknowledgement message. The acknowledgement messages can be transmitted in code form (three-digit number in the form of text characters) or as explicit messages. All information, except the names of sender and receiver, is case insensitive.

The basic size for data packets in TCP connections is the byte. SMTP data is based on the 7-bit ASCII alphabet. Each character is therefore transmitted as an 8-bit byte with the most significant bit as the zero bit.

For practical installation, only a few commands of the SMTP protocol are required. The table below gives a brief overview:

Command	Parameter	Description
HELO	**<domain>**	Welcome *(HELLO)*: Identification of the domain of the Client MTA (sender) for the server MTA (receiver)
MAIL	**FROM:** **<reverse path>**	Initialization of the mail transfer from the sender, identification of the sender **<reverse path>**
RCPT	**TO:** **<forward path>**	- Specification of the recipient: Identifikation of the delivery address **<forward path>**, - multiple recipients can be specified by repeated call
DATA		- Start of message transmission, - Message can consist of a large Number of lines with ASCII text, - End of the message with a single point in a line and followed by CR/LF
QUIT		Termination of the mail transmission and closing the TCP connection
VRFY	**<string>**	Verification *(Verify):* Request to the server MTA to send Back an acknowledge message for the existence oft he recipient named **<string>**
EXPN	**<string>**	Extension *(Expand):* Return the server MTA to extension the address list by the address **<string>**
TURN		Swap the roles (tasks) of the client MTA and the server MTA
NOOP		Empty (no) operation *(NO-OP)* without any effect on the transfer
HELP		Call for help to the receiver to return Helpful information for the sender.

The table below shows various acknowledgement codes and the associated acknowledgement messages from the receiver (in numerical ascending order):

Code	**Acknowledgement message**	**Description**
214	**Help message**	Specification for help
220	**<domain> service ready**	Readiness for mail transfer
221	**<domain> service closing trans-mission channel**	Closing the transmission channel
250	**OK**	- Acceptance of the mail request, - Acceptance of the recipient address, - Confirmation end of the message
251	**Failure reply: user not local, will forward to ...**	Error: wrong delivery address, correct address is available to the recipient, recipient forwards mail
354	**Intermediate reply: Start mail input**	Acceptance of readiness to send for messages to be sent
421	**<domain> service not available**	- Service not installed, - Closing the transmission channel
500	**Syntax error**	Syntax error in command
501	**Syntax error**	Syntax error in parameter or argument
502	**Command not implemented**	unknown command
504	**Command parameter not implemented**	unknown parameter
550	**Failure reply: Mailbox unavailable**	Error: unknown delivery address
551	**user not local please try ...**	- Error: wrong delivery address, - correct address is known to recipient, - recipient recommends forwarding
552	**exceeded storage allocation**	Transmission aborted due to lack of storage space.

8.7.2 Advanced email services

The SMTP protocol is severely limited in its possible uses for current applications:

- It only allows the transmission of 7-bit ASCII characters.
  The use of the extended ASCII code with a width of 8 bits is not possible.

- Executable code or other binary files cannot be transmitted.

- Attachments of files cannot be transmitted, e.g. images and sound files.

For these reasons an extension of the original SMTP standard was made: **Multipurpose Internet Mail Extension MIME** (RFC 2045-2048, new: 2184).

The MIME standard includes the following additions:

- Use of various file formats to support multimedia electronic mail
  (images, movies, audio data, binary data, ...),

- Definition of encodings for the transmission of attachments.