TCP/IP Standard Applications
Telnet - SSH - FTP - SMTP - HTTP

Virtual Terminal, Secure Shell, File Transfer, Email, WWW

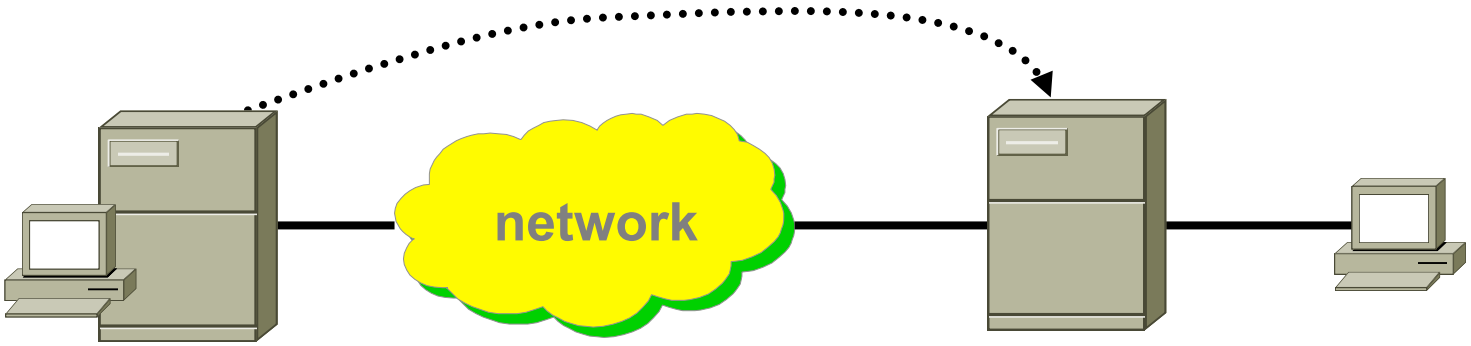
Agenda

- Telnet (Virtual Terminal)
- SSH
- FTP (File Transfer)
- E-Mail and SMTP
- WWW and HTTP

What is Telnet?

- **Telnet is a standard method to communicate with another Internet host**
- **Telnet provides a standard interface for terminal devices and terminal-oriented processes through a network**
- **using the Telnet protocol user on a local host can remote-login and execute commands on another distant host**
- **Telnet employs a client-server model**
  - a Telnet client "looks and feels" like a Terminal on a distant server
  - even today Telnet provides a text-based user interface

Local and Remote Terminals



local terminal

workstation
as remote terminal
with Telnet Client

Host
with Telnet Server

traditional configuration

today's demand: remote login

About Telnet

- **Telnet was one of the first Internet applications**
  - since the earliest demand was to connect terminals to hosts across networks
- **Telnet is one of the most popular Internet applications because**
  - of its flexibility (checking E-Mails, etc.)
  - it does not waste much network resources
  - because Telnet clients are integrated in every UNIX environment (and other operating systems)

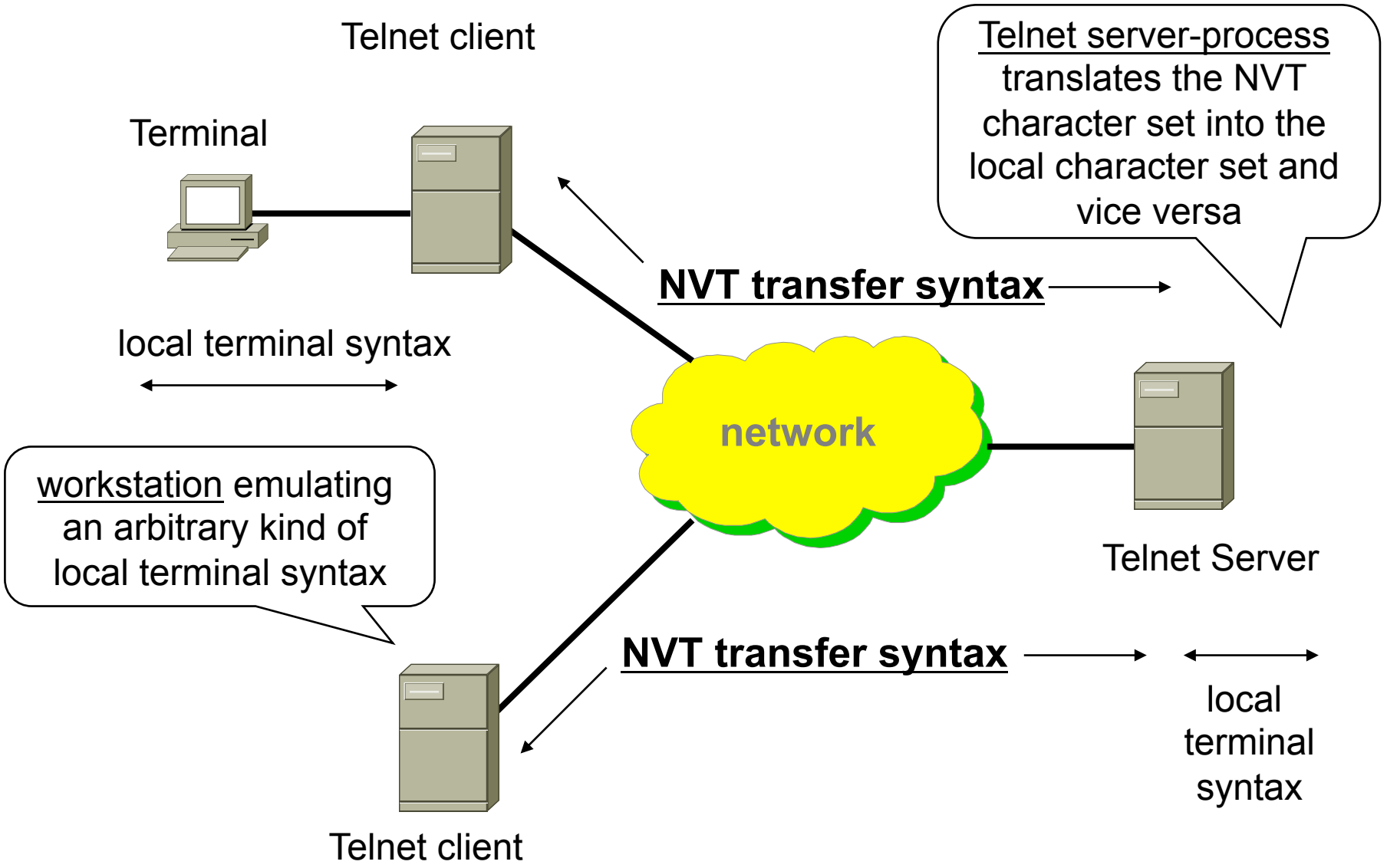
Telnet Basics

- **Telnet is connection oriented and uses the TCP protocol**
- **clients connect to the "well-known" destination port 23 on the server side**
- **protocol specification: RFC 854**
- **three main ideas:**
  - concept of *Network Virtual Terminals* (NVTs)
  - principle of *negotiated options*
  - a *symmetric view* of terminals and (server-) processes

Virtual Terminals

- **a Telnet Client can emulate the behaviour of a wide range of well-known real terminals**
- **internally, each end of a Telnet connection leads to a Network Virtual Terminal (NVT)**
- **an NVT provides a standard, network-wide, intermediate representation of a canonical terminal**
  - consisting of a display (printer) and a keyboard (line-buffered mode) in half-duplex mode
  - Telnet communications rely upon the "language" of NVTs
  - each local device characteristics are mapped to the NVT capabilities

Telnet Client - Server



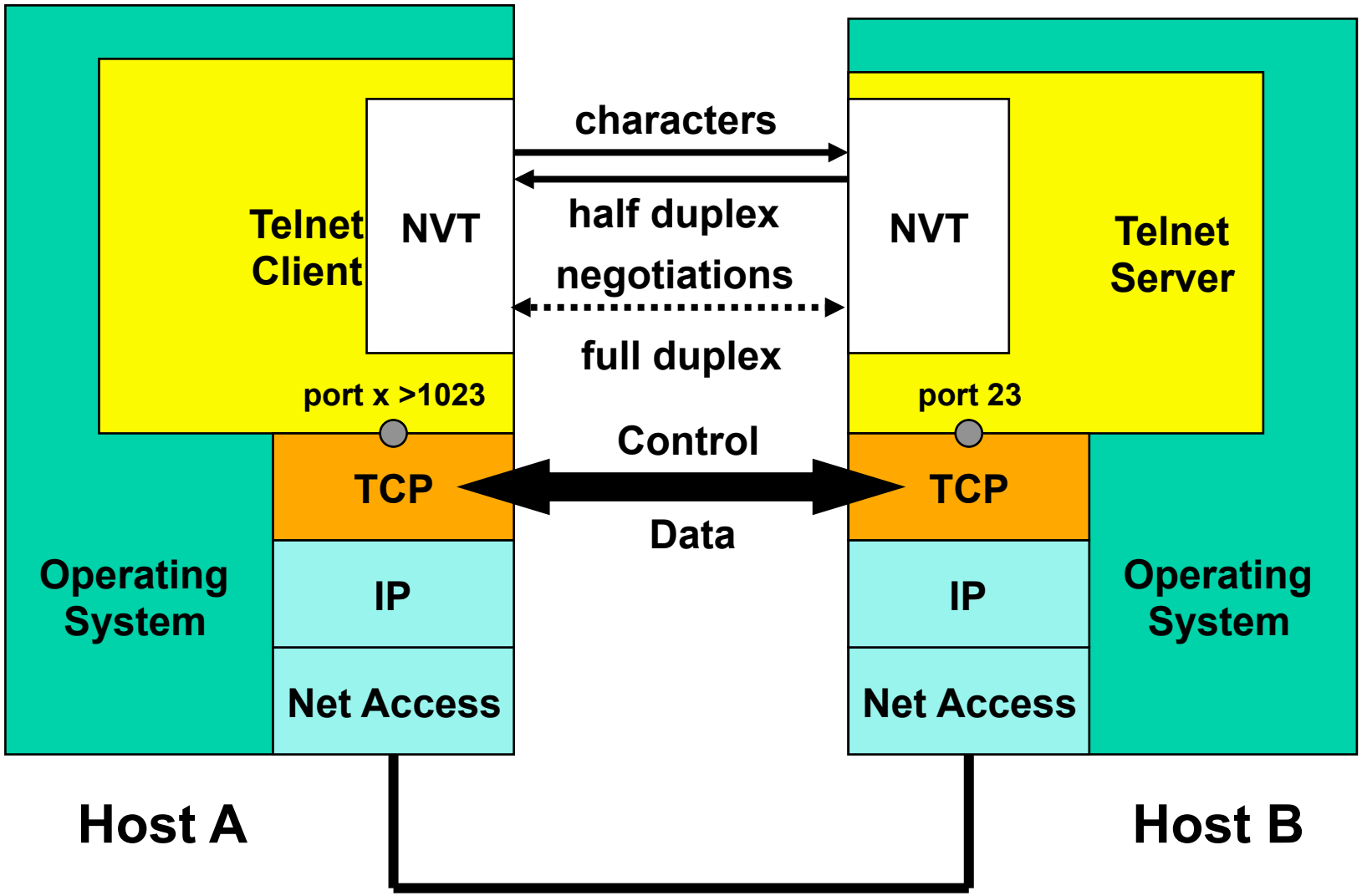
Half-Duplex Connection

- **a Telnet connection "itself" is running full-duplex**
  - e.g. both sides can send negotiation commands or signals at the same time
- **but at the users point of view, NVTs only communicate in a half-duplex way !**
  - to reduce network costs and the number of server interrupts, a Telnet-client accumulates NVT keyboard inputs in a buffer before sending it (e.g. line buffered)
  - on the other side the Telnet-server wants to send all data to the client's printer before the client continues
  - so a kind of token-principle has been specified: the GA-character (Go Ahead) **can** be send to notify the other side that the current sender has finished its transmission

Negotiating Options

- **in order to extend the rather poor capabilities of a NVT, Telnet provides a means for option-negotiating**
  - using commands like DO, DON'T, WILL, WON'T
  - e.g. for full screen mode, specify terminal type, etc...
- **symmetric view: both the server and the client may propose additional options to be used**

Symmetric Telnet Model



NVT's Character Set

- **NVT generally use the 8 bit data format**
- **however, NVT's basic character set is the US ASCII 7-bit code**
- **so an NVT can handle the printable characters with ASCII codes 32-126 plus a small set of control characters:**
  - NULL (NUL) - no operation
  - BELL (BEL) - produces an audible or visible signal
  - Back Space (BS) - moves the print head one character to the left margin
  - Horizontal Tab (HT) - moves the printer to the next horizontal tab stop
  - Line Feed (LF) - moves the printer to the next print line, keeping the same horizontal position
  - Vertical Tab (VT) - moves the printer to the next vertical tab stop
  - Form Feed (FF) - moves the printer to the top of the next page
  - Carriage Return (CR) - moves the printer to the left margin

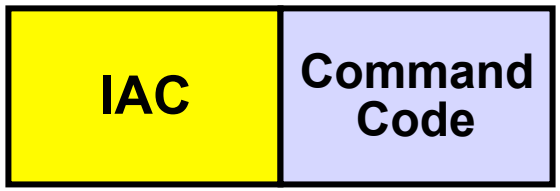
Internal Telnet Commands

- **for options negotiating and signalling purposes Telnet applies special command characters**
- **these commands have bit 8 set (code words 128-255)**
- **Telnet commands are prefixed with a special escape character: IAC - "Interpret As Command"**
  - code word 255
  - IAC is doubled if it appears in the normal data stream (only in the optional 8-bit mode - "IAC stuffing")

Internal Telnet Commands

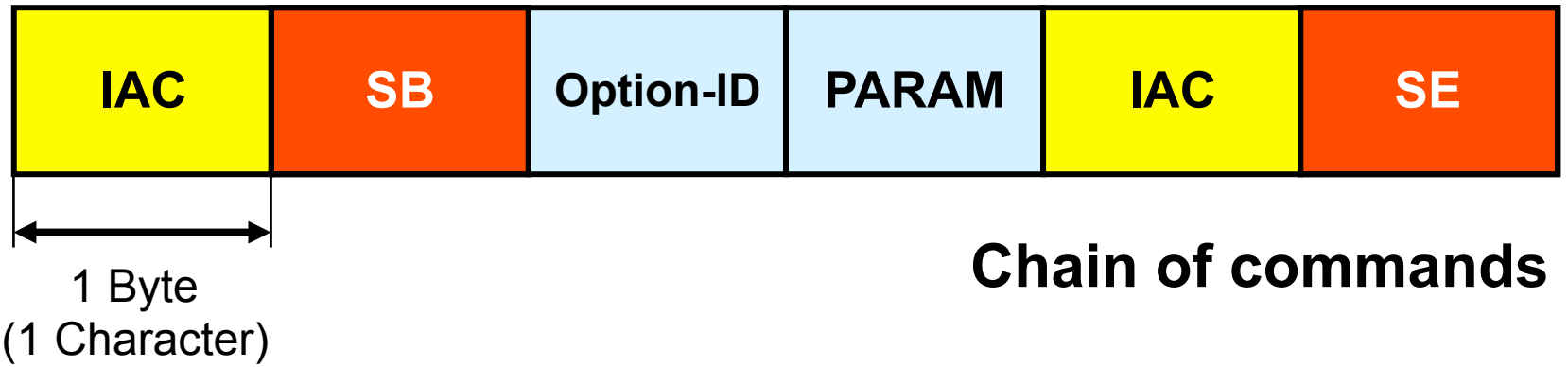
- **all communication between client and server is handled with internal commands**
- **each command has 2 or 3 bytes length**
  - first byte: IAC
  - second byte: command code
  - possible third byte: referenced option when negotiating
- **the chain of commands can be even longer in case of sub-negotiating**
  - indicated with the command code SB (Subnegotiation Begin)
  - closed with the command code SE (Subnegotiation End)

Possible Internal Command Formats



Standart Formats

Reference option when negotiating

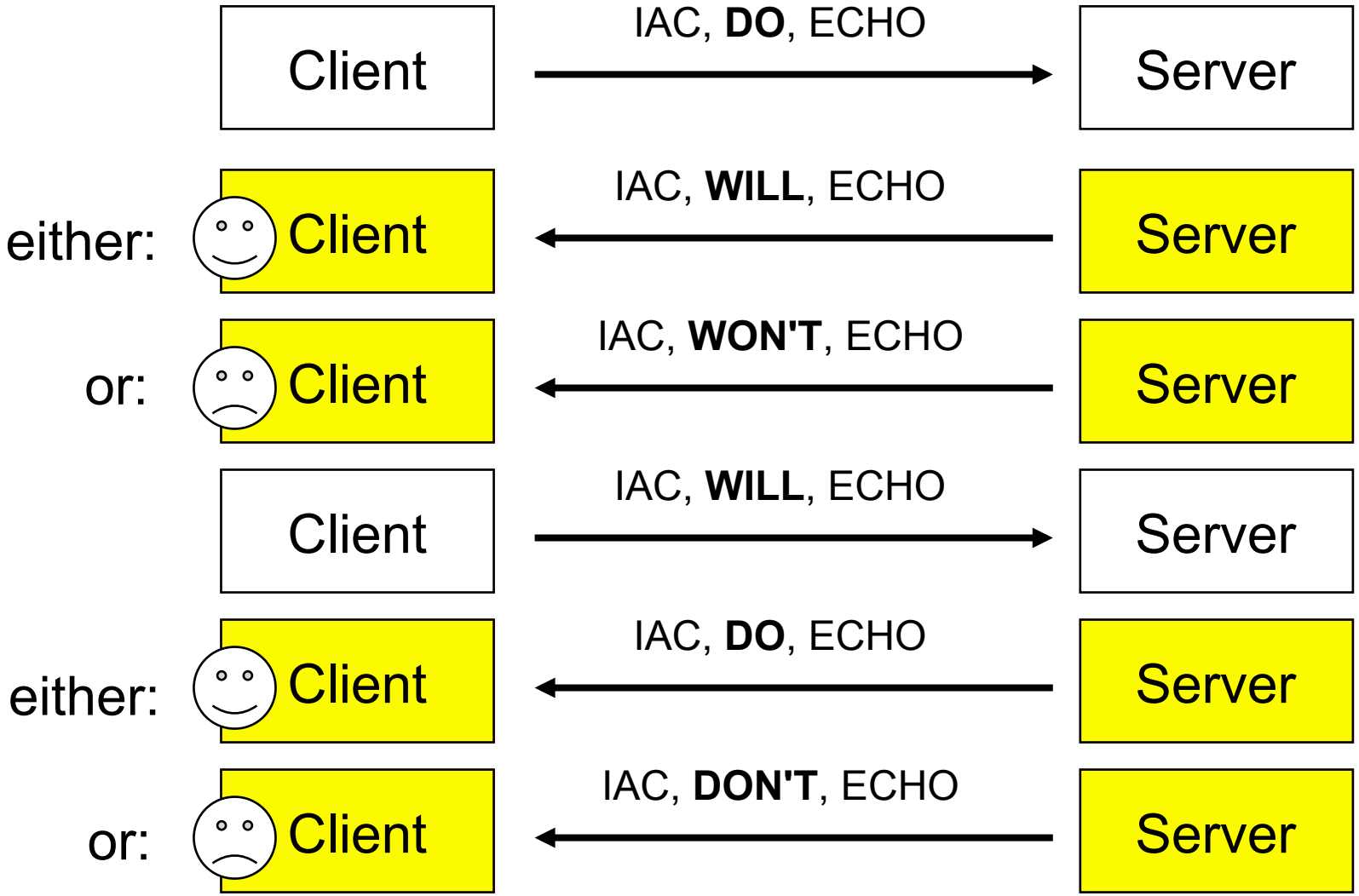


Chain of commands

Internal Telnet Commands - Overview

	SE	240	End of Subnegotiation
	NOP	241	No Operation
	DM	242	Data Mark (part of the Synch function)
	BRK	243	NVT character break
	GA	249	Go Ahead ("Token" for half duplex mode)
	SB	250	Begin of Subnegotiation
negotiation commands	**WILL**	251	Sender wants to enable an option
	WON'T	252	Sender do not want to enable an option
	DO	253	Sender asks Receiver to enable an option
	DON'T	254	Sender asks Receiver to not enable an option
	IAC	255	Interpret As Command

Command Examples for Negotiation



Important Telnet Options - Overview

0	Transmit Binary
1	Echo
3	Suppress Go Ahead
5	Status
6	Timing Mark
8	Output Line Width
9	Output Page Size
24	Terminal Type
35	X Display Location
39	Telnet Environment Option

Important Telnet Options (1)

- **Transmit Binary (Code 0)**
  - toggles from 7-bit ASCII code to 8-bit binary code with IAC stuffing
- **Echo (Code 1)**
  - received data characters will be echoed back to the sender
  - by default local echo (character on screen is echo of client keyboard) is enabled
- **Suppress Go Ahead (Code 3)**
  - toggles from the default half-duplex mode into full-duplex
- **Status (Code 5)**
  - verify the current status of remote Telnet options

Important Telnet Options (2)

- **Timing Mark (Code 6)**
  - causes the a time stamp to be inserted inside the data stream (for synchronisation purposes in full-duplex mode)
- **Terminal Type (Code 24)**
  - to signal some specific terminal type to be used
    - DEC VT-100, IBM 3270
- **Extended Options List (Code 255)**
  - if there is a demand for more than 256 Telnet options, this option can be used to negotiate the availability of an extended option list

Important Telnet Options (3)

- **Telnet Environment Option (Code 39)**
  - enables the server to use its client's environment variables
- **Output Line Width (Code 8)**
- **Output Page Size (Code 9)**
- **X Display Location (Code 35)**

Basic Set of Standard Functions

- **to ease the compatibility of different implementations**
  - a set of standard functions have been specified (= most important functions)
  - each of these commands initiates the processing of a well defined control function

IP	**244**	**Interrupt Process**
AO	**245**	**Abort Output**
AYT	**246**	**Are You There?**
EC	**247**	**Erase Character**
EL	**248**	**Erase Line**
SYNCH	**----**	**Synchronization**

Standard Functions - Explanation (1)

- **IP - Interrupt Process**
  - invokes a system function to suspend, interrupt, abort or terminate the operation of the (remote) process
- **AO - Abort Output**
  - forces the remote system to finish its output, even if there is any outstanding data
- **AYT - Are You There**
  - requires the remote system to send an optical (printable) or acoustic ("beep") signal to indicate that this system is still up and running
- **EC/EL - Erase Character/Line**
  - this function is typically used to edit keyboard input

Standard Functions - Explanation (2)

- **SYNCH - Synchronize**
  - processes in remote systems are sometimes hard to control because some control signals might be buffered anywhere between the sender and the receiver
    - e.g. caused by the networks flow control
  - the Telnet "Synch" mechanism consists of a TCP Urgent notification coupled with the Telnet DM (Data Mark) command
  - on receiving any data stream with the TCP-Urgent data bit set, a server discards all buffered data except commands
  - the Telnet DM-command signals that the desired commands have been already occurred and the server can return with normal processing the data stream

Synchronised Commands

- **the Telnet SYNCH function is applied on the most essential basic functions:**
  - AYT, AO, IP and BRK
- **that is, these characters are send in TCP segments with the Urgent data bit set, followed by a Telnet DM command**

Security Issues

- **Telnet-clients are able to connect to many server-ports (if not closed for Telnet connections)**
  - port 25 (SMTP) can be used for faked E-Mails
  - port 6000 (X-Window) can be monitored to catch window-contents, passwords, jammed for Denial of Service (DoS), ... (if not protected using xhost or magic cookies)
  - port 80 (HTTP) can also be a target for DoS; recently, the NT-webserver IIS could be easily crashed via port 135 (and others)
- **Telnet does not encrypt passwords -> sniffers !!!**
  - so never give telnet users root privileges (some operating systems disallow remote root-logins anyway)
  - use secure shell (SSH) for security reasons

Relevant RFCs

- RFC 854 - Telnet Protocol Specification
- RFC 855 - Telnet Option Specifications
- RFC 856 - Telnet Binary Transmission
- RFC 857 - Telnet Echo Option
- RFC 858 - Telnet Suppress Go Ahead Option
- RFC 859 - Telnet Status Option
- RFC 860 - Telnet Timing Mark Option
- RFC 861 - Telnet Extended Options - List Option
- RFC 1184 - Telnet Linemode Option

Agenda

- Telnet (Virtual Terminal)
- SSH
- FTP (File Transfer)
- E-Mail and SMTP
- WWW and HTTP

SSH Basics

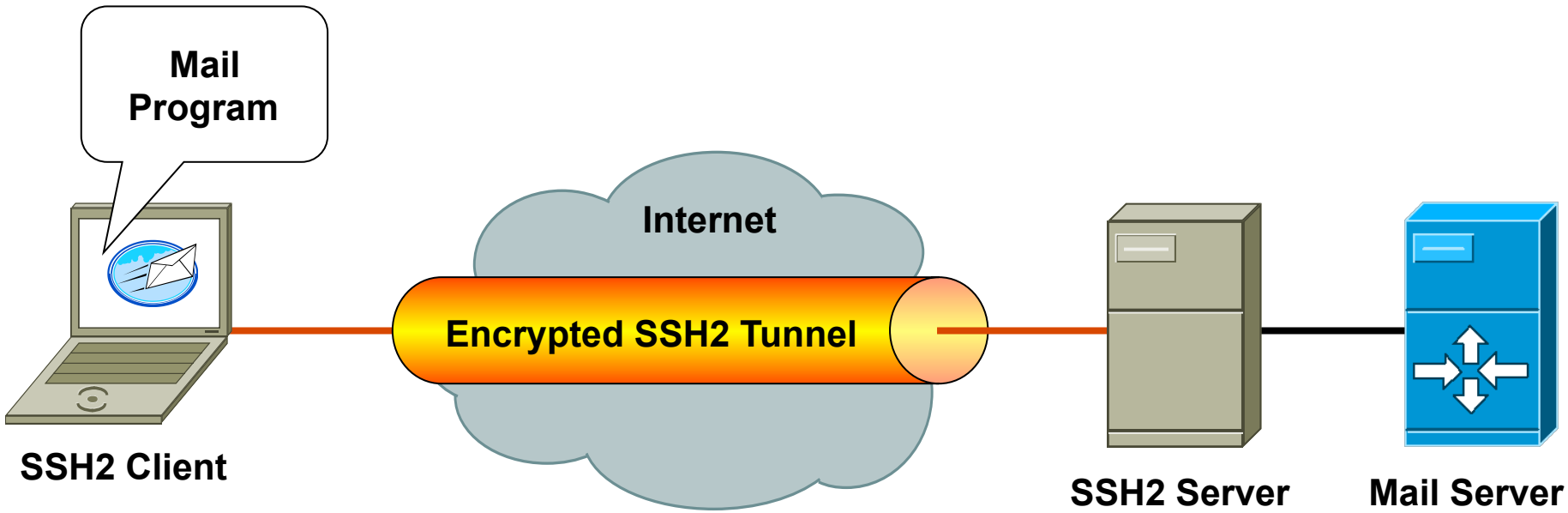
- **Secures connections over the Internet**
- **Encrypting all transmitted confidential data**
  - Passwords
  - Binary files
  - Administrative commands
- **Two versions of Secure Shell (not compatible)**
  - Secure Shell Version 1(SSH1 or SSH)
  - Secure Shell Version 2 (SSH2 or SecSH)
- **De-facto standard**
- **Client-server protocol**

SSH Basics

- **Solve two most acute problems in the Internet**
  - Secure remote terminal logins
    - ssh -l user-name machine-name
  - Secure remote command execution
    - ssh machine-name/path to exe-file
  - Secure file transfers
    - scp file user-name@machine-name
  - Port forwarding
    - ssh -L 3002:hostB:119 hostB
- **Tunnels TCP sessions over encrypted Secure Shell connection**
  - Secure the communication of other applications and protocols without modifying the applications

Principle

E-Mail Security through SSH2 Tunneling



Encryption

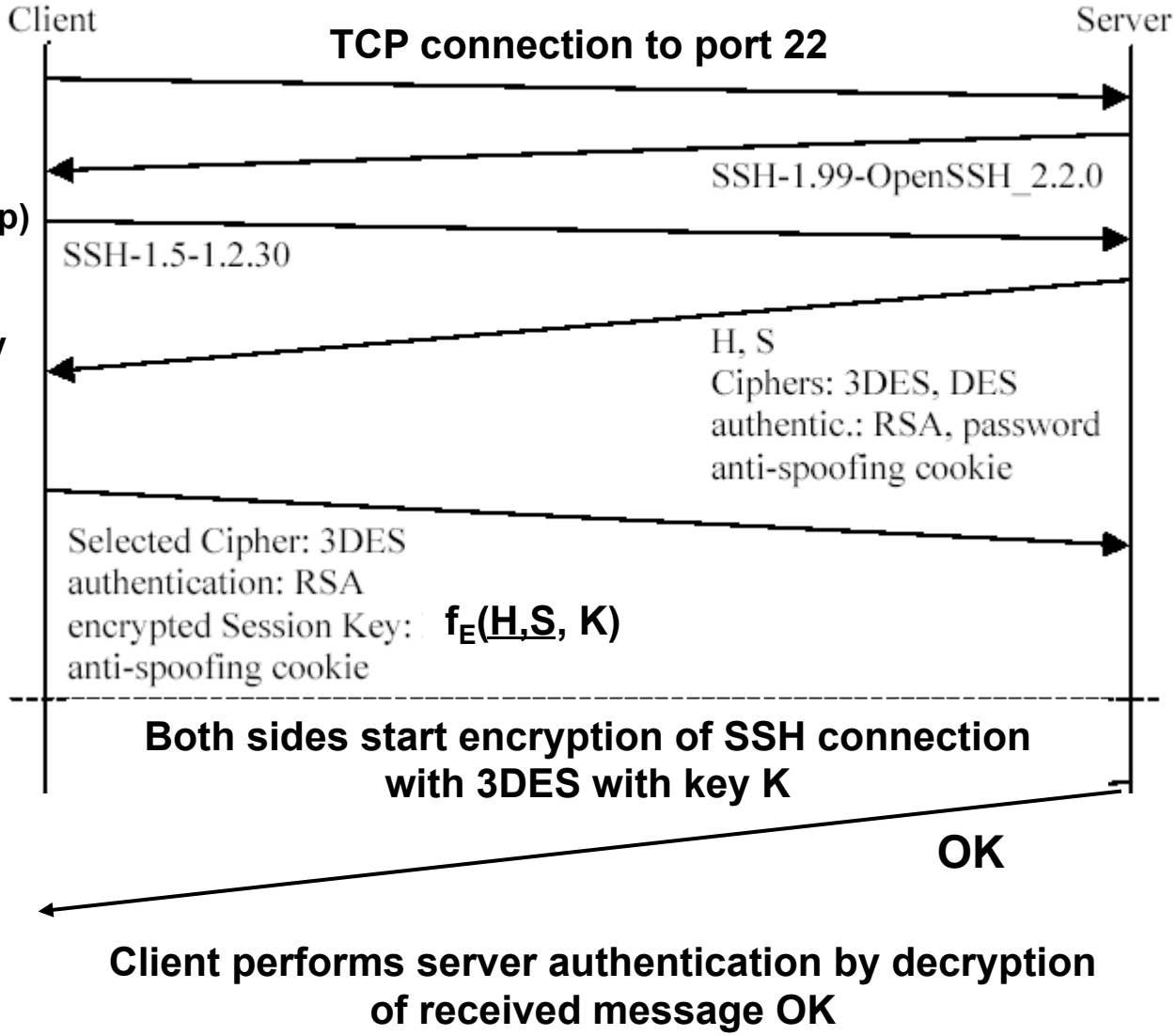
- **Support of the strongest available encryption algorithms**
  - 3DES
  - CAST-128
  - Twofish
  - AES
    - Advanced-Encryption-Standard (US)
    - 128-bit key!

Method	SSH1	SSH2
DES	X	-
3DES	X	X
IDEA	X	-
Blowfish	X	X
Twofish	-	X
Arcfour	-	X
AES	-	X
Cast128-cbc	-	X

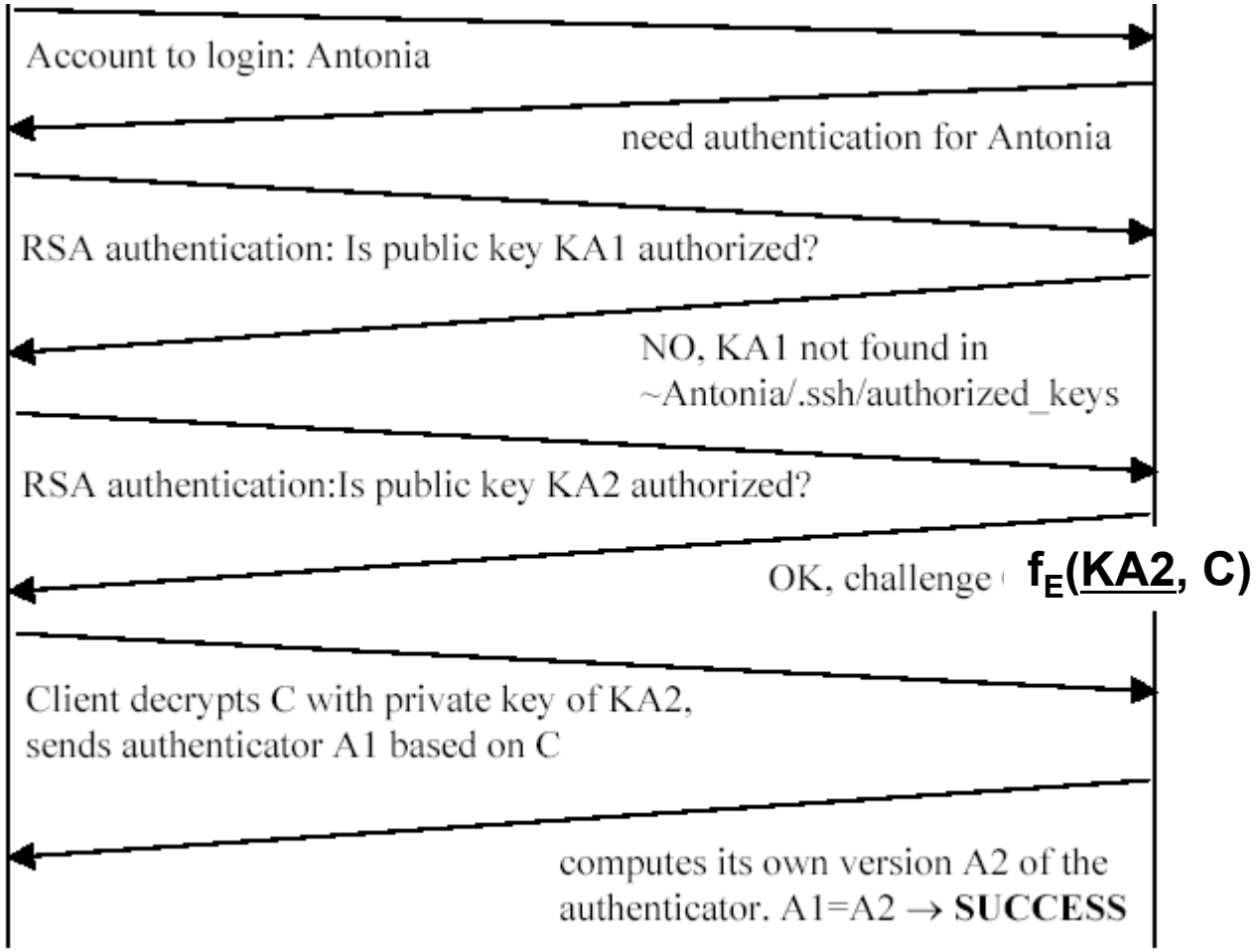
SSH (v1) Procedures 1

H … public-key RSA Host (generated by ssh server setup)
S … public-key RSA Server (generated by connection start and changed periodically for perfect forward secrecy)

client checks if H, S already know in file ~User/.ssh/known_hosts;
if yes -> ok
if no -> ask user



SSH (v1) Procedures 2



SSH1 vs. SSH2

- **Two entirely different protocols**
- **SSH1 uses server and host keys to authenticate**
- **SSH2 only uses host keys**
- **SSH2 encrypt different parts of the packet**
- **SSH2 is a complete rewrite of the protocol**
- **SSH2 is more secure**
- **Where to get:**
  - OpenSSH -> http://www.openssh.com/
    - ssh, scp, sftp, sshd, stfp-server
  - PuTTY -> http://www.chiark.greenend.org.uk/~sgtatham/putty/
    - Telnet and SSH client
  - SSH Tectia -> http://www.ssh.com/

Agenda

- Telnet (Virtual Terminal)
- SSH
- FTP (File Transfer)
- E-Mail and SMTP
- WWW and HTTP

File Transfer Protocol FTP (RFC 959)

- **the way information is stored depends on the architecture of the underlying system**
  - hardware- and software-architecture (HW - processor; SW - operating system)
  - datatypes and coding styles
  - file organization and access methods
- **two approaches possible for exchanging files between different systems**
  - definition of virtual files and translation to real files
  - reduction: extract some few fundamental properties from many individual properties

Virtual File Approach

- **all possible representations must be considered**
- **translators from real to virtual filesystems and vice versa must be implemented**
  - complex and difficult to realize
  - advantages: operating systems working with virtual file-stems can easily support a variety of real filesystems
- **examples**
  - ISO FTAM protocol (layer 7)
    - FTAM (File Transfer, Access and Management) also allows to manage a remote filesystem
  - Linux Kernel
    - using an internal virtual filesystem it was easy to implement support for HPFS, NTFS, FAT, OS/2, System V, UFS, and other filesystems

Reduction Approach

- **based upon common fundamental properties of each filesystem**
  - data types, file organization, file ownership and access authority, symbolical names for file identification, I/O-operations, etc.
  - only fundamental views and manipulation operations
    - easy to implement and powerful
  - no translation necessary between different systems
    - application itself is responsible for the appropriate data format
- **example: FTP**

Difference: FTP - File Server OS

- **FTP: *Sharing by File Transfer***
  - files are copied and forwarded to the local system; the original file remains unchanged
- **File Server OS: *Online Sharing Systems***
  - allows multiple users to share a file over a network
  - files from a fileserver can be accessed and manipulated like local files
  - examples: Novell File Server, Sun NFS, IBM Lan Manager

FTP-Dimensions for Filetransfer

- **data-representation (dimension datatype):**
  - ASCII 7-bit in 8-bit NVT to exchange text between arbitrary systems
  - EBCDIC 8-bit for IBM to IBM transfer
  - IMAGE (8-bit binary) to exchange binary data between similar (compatible) systems
- **file-organization (dimension filetype):**
  - file structure (strings of bytes, end marked by EOF)
  - record structure (list of records, end of each marked by EOR)

    EOF and EOR are represented by sequence of 2-bytes: hexFF and hex01 (EOR) | hex02 (EOF) | hex03 (EOR+EOF) plus bytestuffing if hexFF appears within the (source) data stream

FTP-Dimensions

- **transfer type (dimension transmission mode):**
  - stream ... data is transmitted as continuous bit stream without being modified; only EOF and EOR are represented as an appropriate 2-byte sequence
  - block ... data is divided in uniquely distinguished blocks; EOR marks end of block, EOF marks end of file

    block-mode allows applications to implement restart-mechanisms (to be used in case of transmission errors)
  - compressed ... data is compressed-> sequences of same characters are transmitted only once; additionally a replication counter must be transmitted which tells the receiver how often this sequence occurs

FTP-Principles 1

- **FTP uses client-server communication principle**
- **client-server communication maintains 2 TCP connections**
  - control signals use the well known port 21
  - datastream is connected to the well known port 20 of the server (except passive mode is requested)
- **using TCP means: FTP needs no additional error recovery mechanisms to protect the data**
- **file access protection is done via login-procedure**
  - login name
  - password

FTP-Principles 2

- **after connection establishment of the control connection the client protocol interpreter (PI) and the server PI communicate on the control channel using the NVT format**
- **PI is responsible for**
  - translating the local syntax into the NVT syntax
  - issuing an appropriate action in the underlying OS (e.g. DOS command DIR -> UNIX command LS)
- **control connection provides commands from the client to the server and acknowledgements in the other direction**

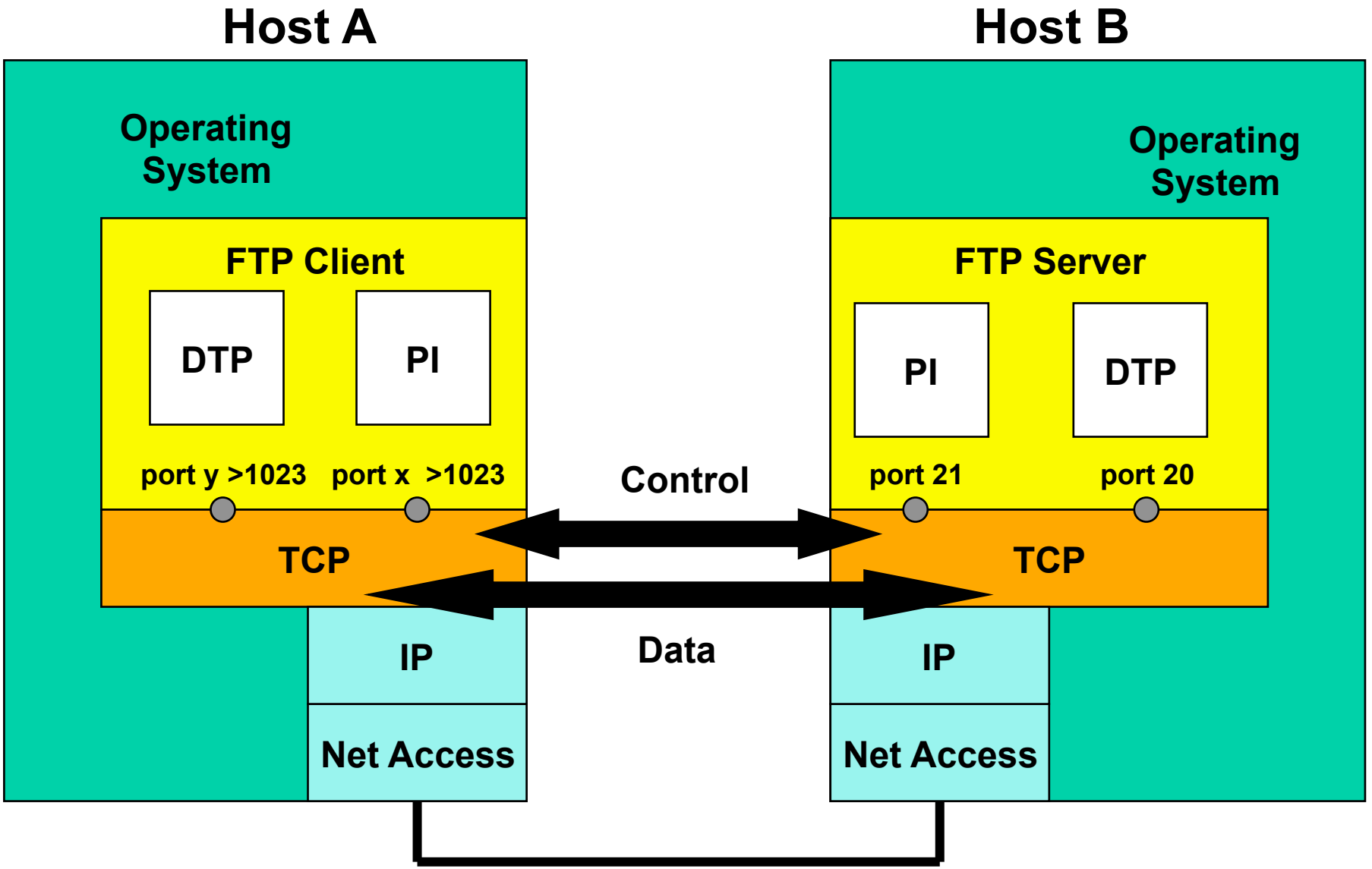
FTP-Principles 3

- **if a command issues a data transfer**
  - a client DTP (Data Transfer Process) and a server DTP are started to maintain a separate TCP- connection
- **the separate TCP connection for date transfer can be established in two ways**
  - the client specifies via control connection a portnummer to which the server setups a TCP connection from port 20 (active mode, default mode)
  - the client requests via control connection passive mode and receives a new port number (> 1023) from the server to which the client establishes the separate TCP connection (passive mode; firewall-friendly)

FTP-Principles 4

- **all data transmission flows over this channel**
- **at the end this connection is closed and the DTPs terminate**
- **this procedure is repeated for each data transmission**
  - half duplex !

FTP Internal Processes



Control Commands 1

- **commands of the control connection from the client to the server (NVT-format):**

  Login Procedure:

  - USER ..... provides username for login
  - PASS ..... provides password of the user;
    NOTE: transmitted in plain text !!!

  Directory Navigation/Creation:

  - LIST ..... list the directory content
  - CWD ..... change the directory
  - CDUP ..... change to the upper directory level
  - MKD ..... create directory
  - RMD ..... remove directory

Control Commands 2

FTP Service :

- RETR ..... load file
- STOR ..... send file
- DELE ..... delete file
- RNFR .... rename from (changing filenames)
- RNTO .... rename to (changing filenames)
- DECE .... deletes files on the server
- APPE ..... append to data to a file
- ALLO ..... allocate memory for files on the server
- NOOP .... no operation; issues OK message from server
- ABOR .... signals server to abort previous commands

Control Commands 3

- REIN ..... re-initialization; client DTP is terminated, connection to the server is still remaining
- QUIT ..... Logout

Transfer Parameter:

- MODE ..... determine transmission mode
- STRU ..... determine file structure
- STAT ..... show the connection state
- TYPE ..... specification of a specific data format (binary, text ASCII/EBCDIC)
- PORT ..... tell the socket for the data connection (forked server: only the initial announcement connection uses the well known port 20)
- PASV …. request passive mode

Control Commands 4

- **all commands contain the necessary arguments**
  - username, password
  - socket-ID, port-id
  - filename, directory
  - datatype:
    - ASCII, EBCDIC, Image
  - file structure:
    - file or record
  - transmission mode:
    - stream, block or compressed
- **and are completed with CR and LF**

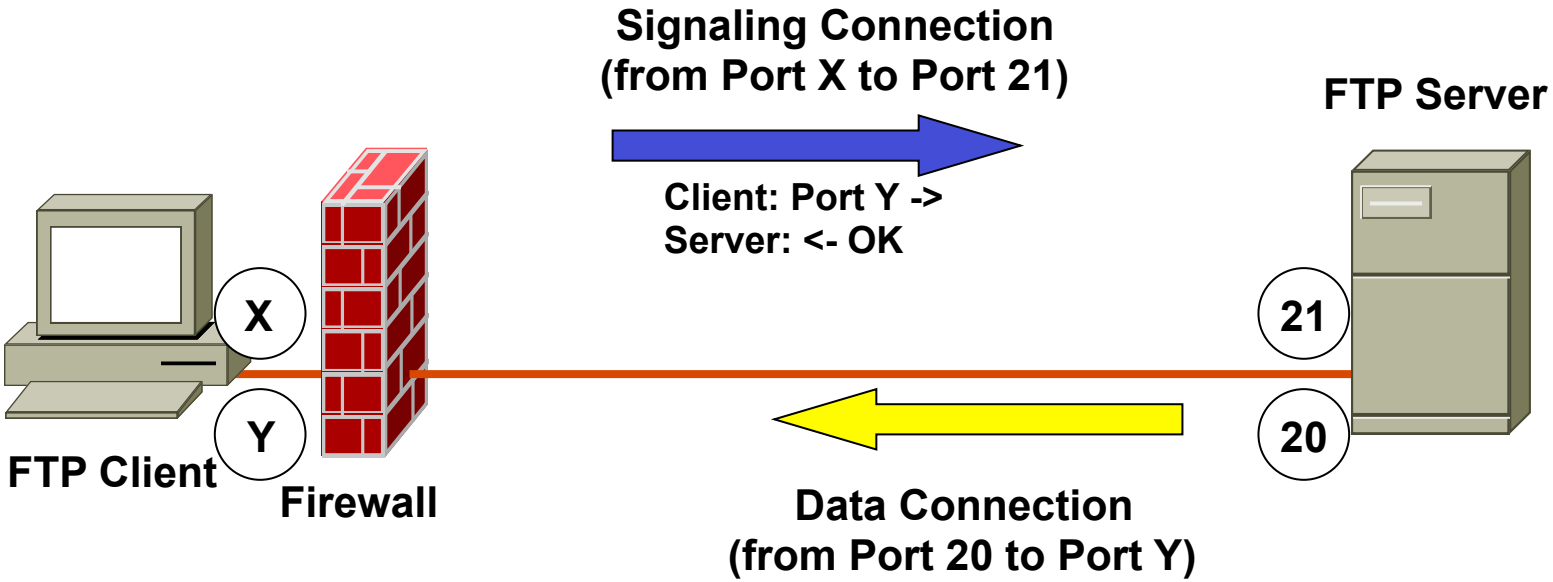
Acknowledge Messages

- **acknowledge types of the control connection from the server to the client (NVT-format):**
  - 220, service ready, CR, LF
  - 331, user name OK, need password, CR, LF
  - 230, user logged in, proceed, CR, LF
  - 200, command OK, CR, LF
  - 150, file status OK, opening data connection, CR, LF
  - 226, closing data connection, CR, LF
  - etc.....
- **acknowledges are printed without further processing**
  - text messages for the user
  - numbers allow easy integration in programs

Acknowledge Coding

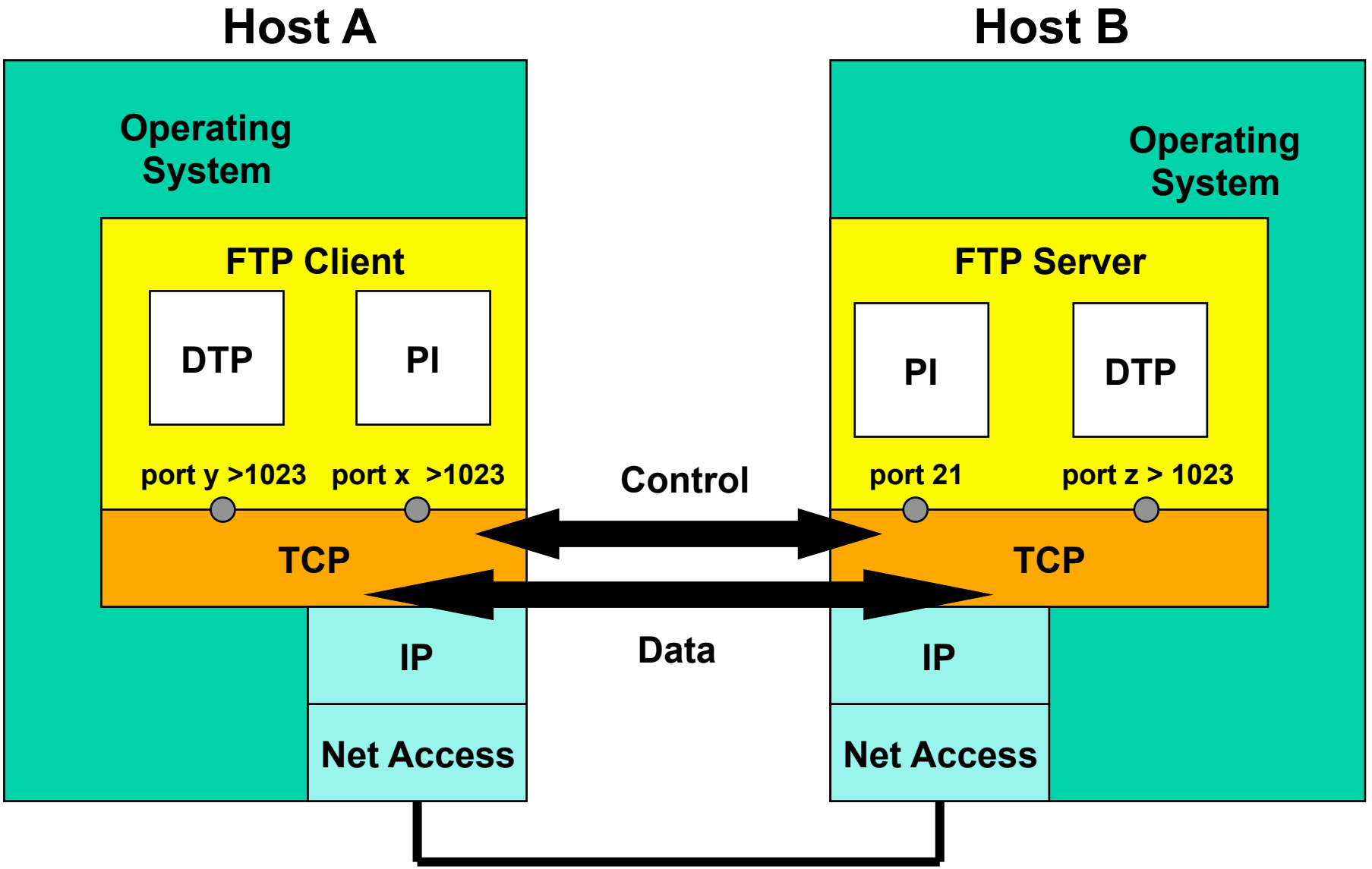
- <1bc> ... premature positive-acknowledge
- <2bc> ... completion-positive-acknowledge
- <3bc> ... meantime positive-acknowledge
- <4bc> ... transient negative-acknowledge
- <5bc> ... permanent negative-acknowledge
- <a0c> ... concerns syntax
- <a1c> ... concerns commands questioning information
- <a2c> ... concerns state of connection
- <a3c> ... concerns commands for identification
- <a5c> ... concerns file system commands
- <ab_> ... detailed acknowledge information

Operation Mode - Classic

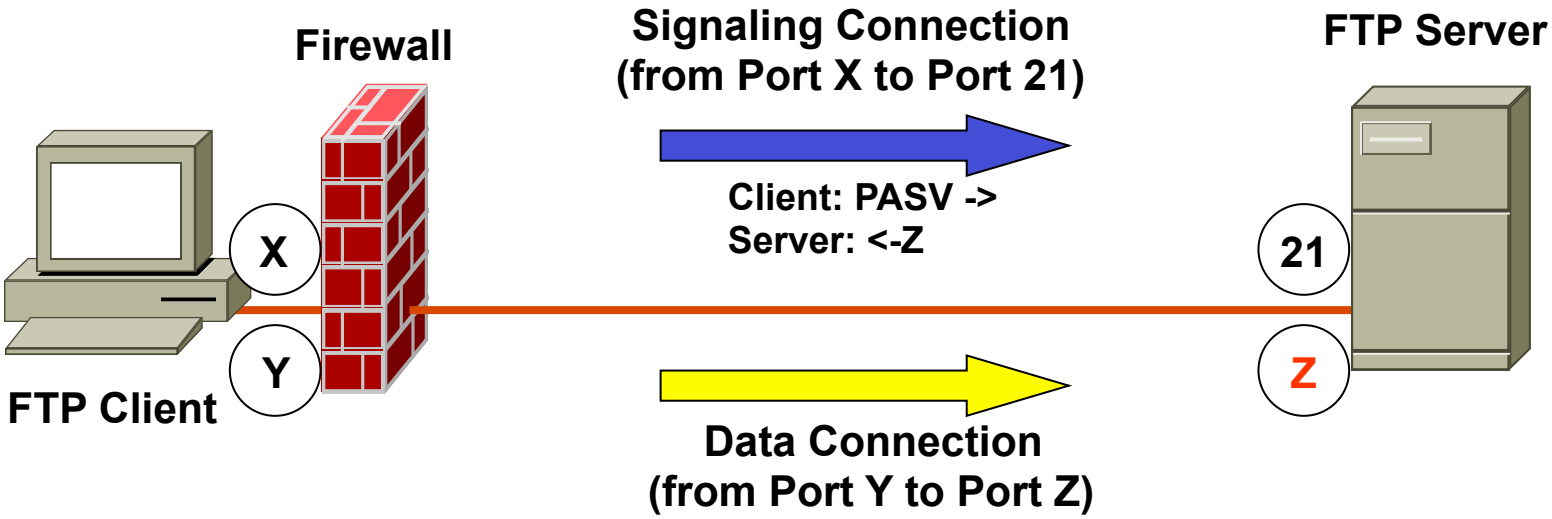


- **Firewall problems**
  - Blocks all incoming connections
- **Old Mode**

FTP Internal Processes (Passive Mode)



Operation Mode - Passive



- **Only outbound connections**
  - No Firewall problems
  - see RFC 1123, 1579, 1635
- **Best mode in secure environment**

User Interface

- **many FTP client software support the following commands through the user interface**
  - open ..... open a FTP connection to a server
  - user ..... announce a new user
  - dir, ls ..... show the directory content
  - pwd ..... show current directory
  - cd ..... change current directory
  - lcd ..... change local directory !
  - binary ..... switch into the image mode
  - text ..... switch into the text-mode (ASCII/EBCDIC) (default?)

Further User Commands

- delete ..... delete a file on the remote system
- get ..... receive a file from the server
- put ..... send a file to the server
- rename .... rename a file
- mget ..... receive multiple files from the server
- mput ..... send multiple files to the server
- mkdir ..... create a directory
- rmdir ..... remove a directory
- exit/quit ... close the connection to the server
- status ..... show the connection state
- ? ..... give help

NOTE: all commands relate to the remote filesystem (filesystem of the server); some commands have local meaning if preceded by a "l"

Agenda

- Telnet (Virtual Terminal)
- SSH
- FTP (File Transfer)
- E-Mail and SMTP
- WWW and HTTP

What is E-Mail ?

- **E-Mail (or "email") is the most widely used Internet application**
  - Note: email was TCP/IP's key to success: developers wrote RFCs and exchange them quickly via email
- **user can communicate with each other**
  - on the same machine or across a network
- **using a mailbox principle**
  - a sender does not require the receiver to be online nor the recipient to be present
  - a user's mailbox can be maintained anywhere in the Internet on a server

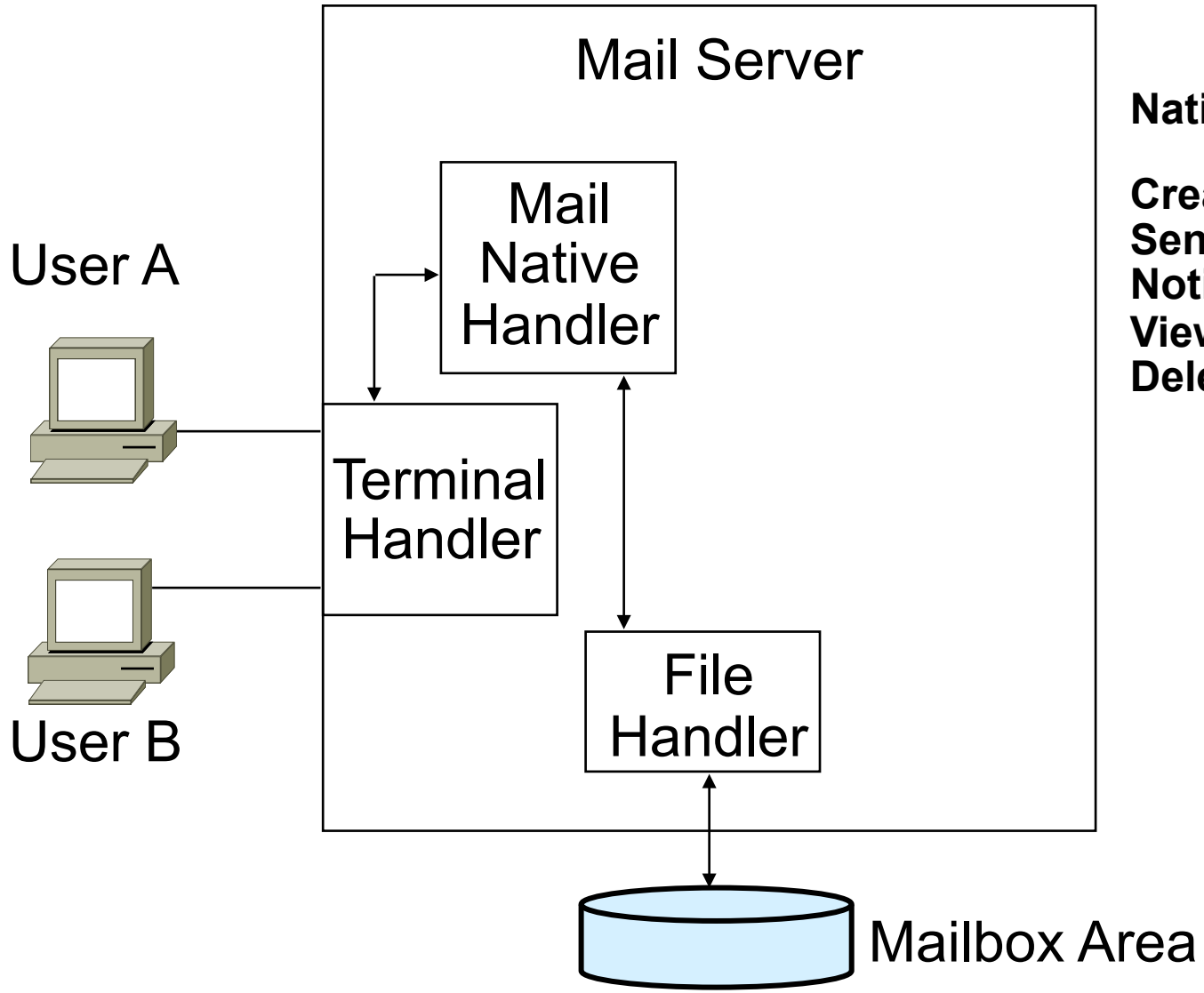
History

- **Electronic Mail has been invented in 1972 by Ray Tomlinson (Note TCP: 1974)**
- **initially started as a simple service that copied a file from one machine to another and appended it to the recipient's "mailbox" file**
- **problems to cope:**
  - several exchange techniques
  - several machine-dependent character sets
  - several mail content formats
  - demand for multi-media extensions
  - demand for encryption
- **1982: standardized mail format (RFC 822)**

The "Electronic Mail System" (EMS)

- **there are several implementations of an EMS**
- **though "Internet-Mail" (using SMTP) is the most popular; standardized by IETF**
  - rather obsolete: Unix to Unix Copy (UUCP)
- **every user owns his own mailbox where he receives and stores messages from other users**
- **every user can be uniquely identified by an email-address**
- **outbound mails are intermediately stored using spooling-ressources**
  - in case of non-standalone EMS

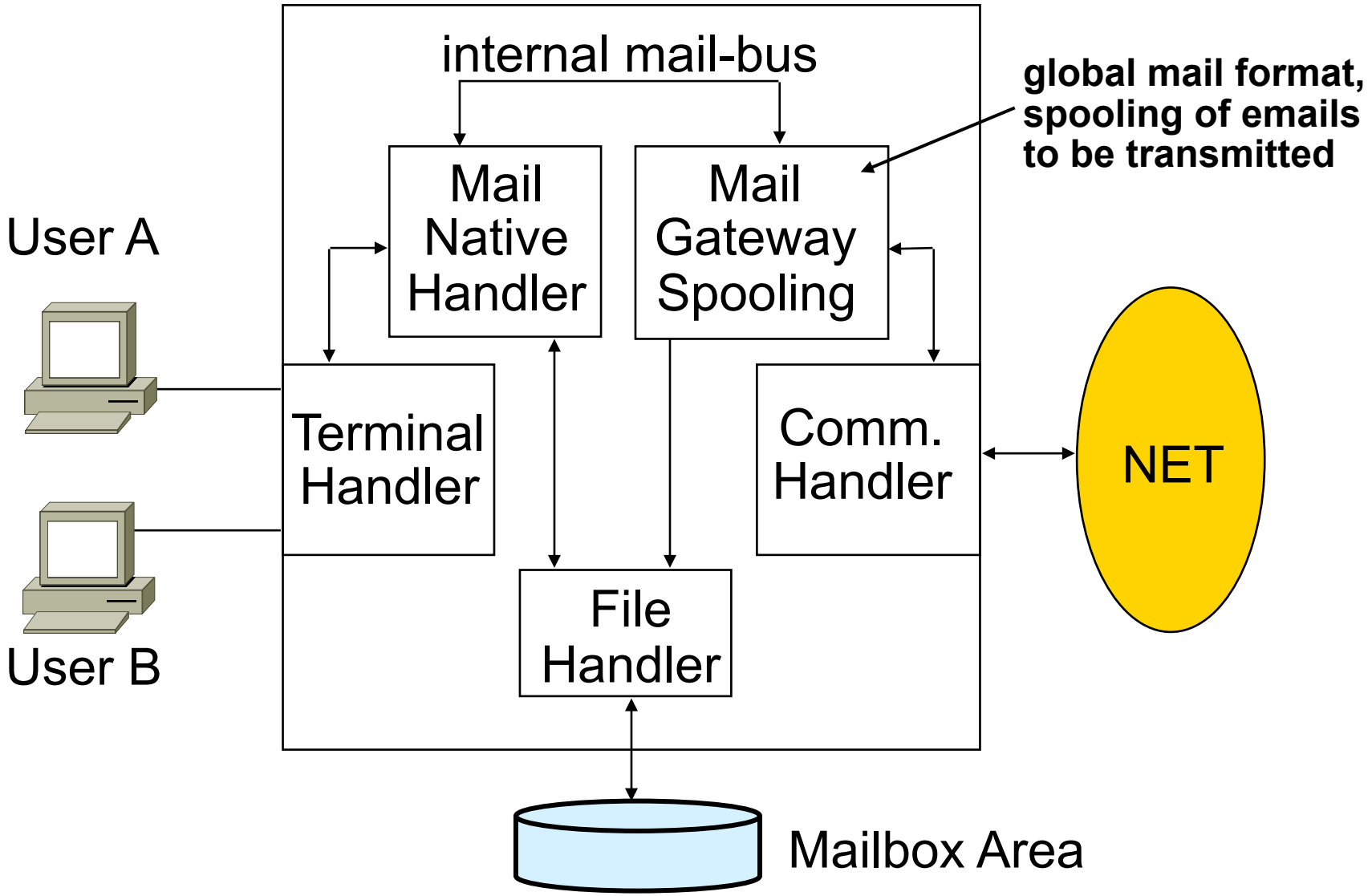
General EMS-Model Standalone



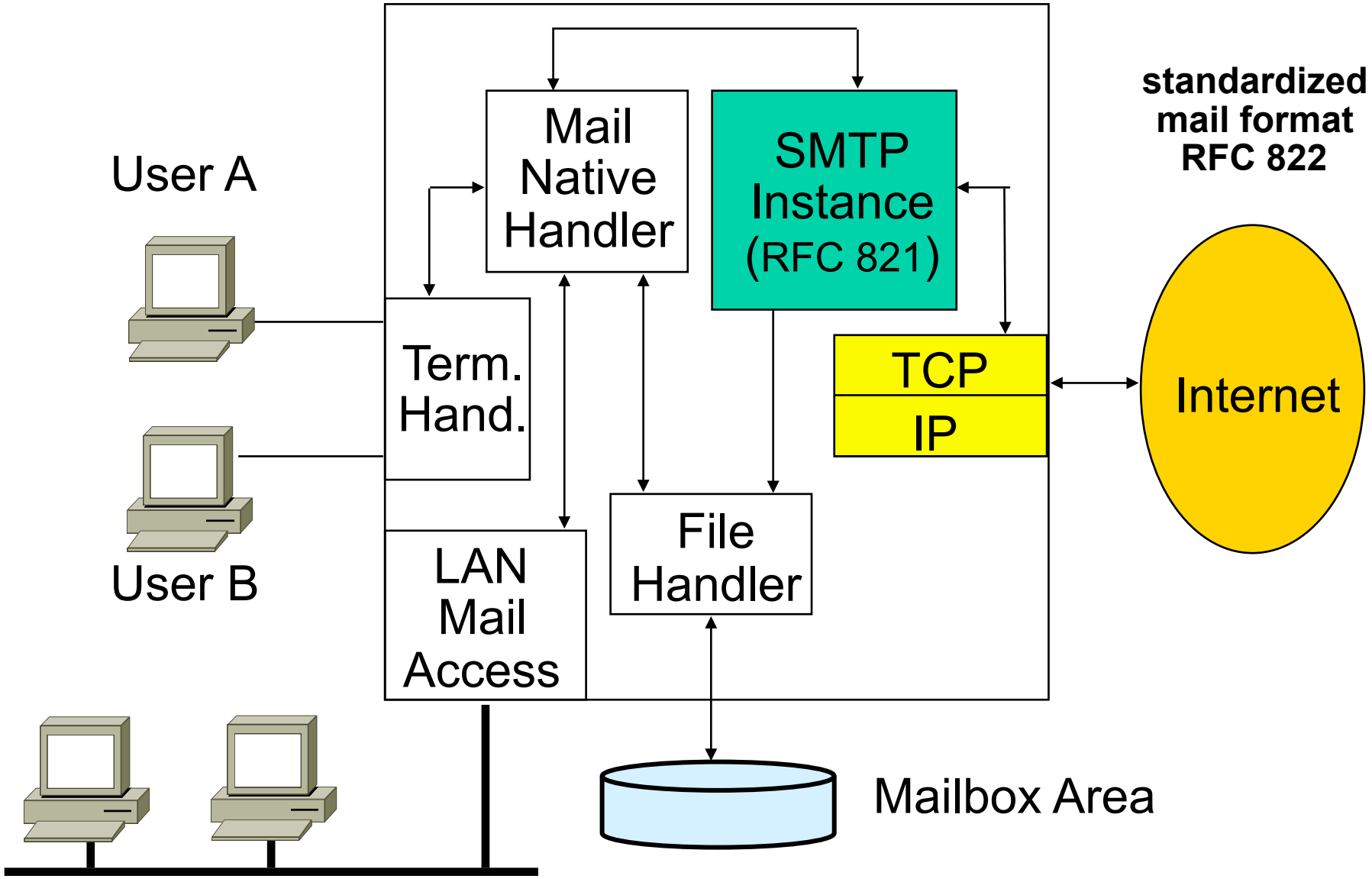
Native mail format

Create Mail
Send Mail
Notify User
View Received Mail
Delete Mail

General EMS-Model Network



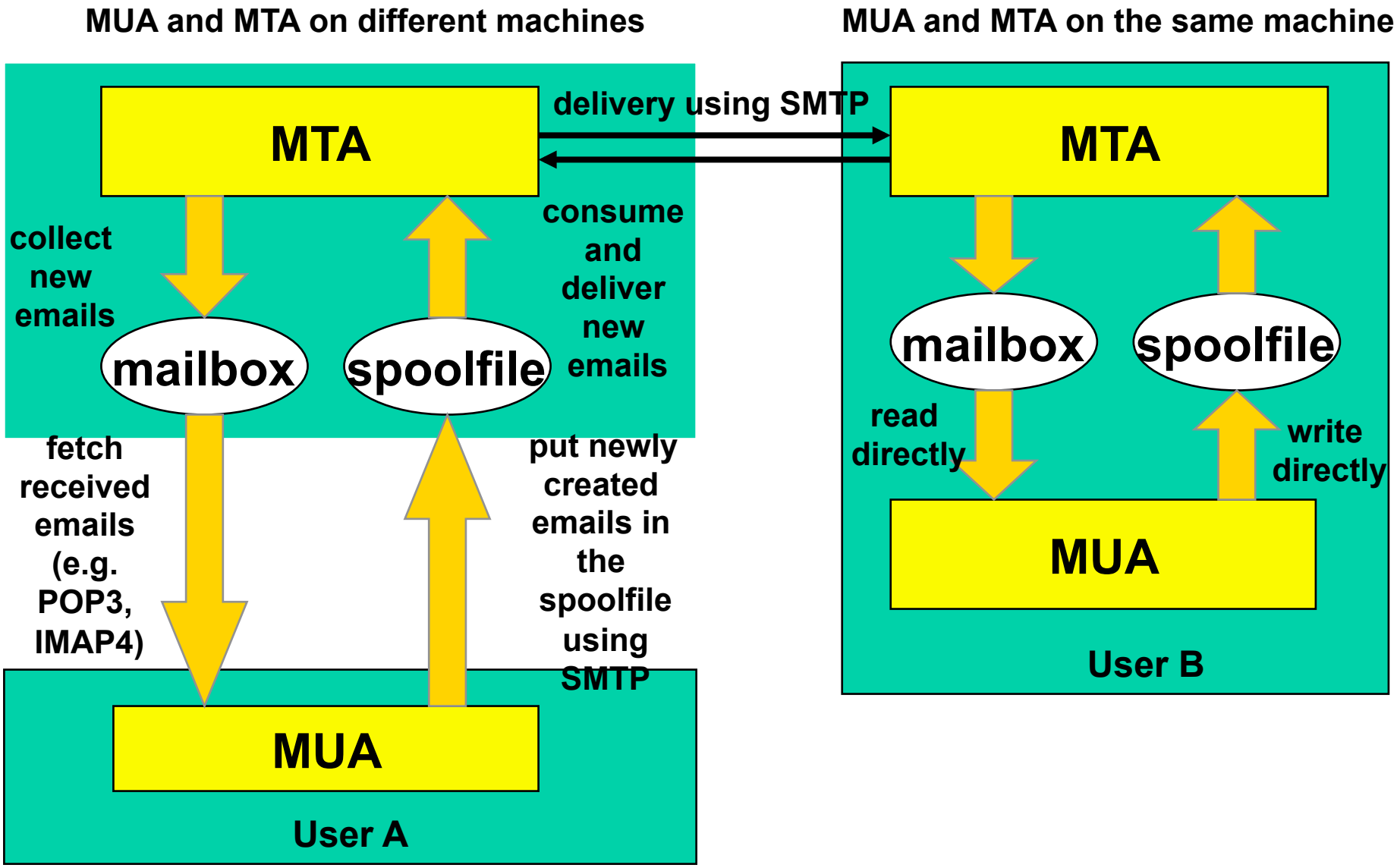
EMS-Model using SMTP



Basic Components

- **Mail User Agent (MUA)**
  - program to read and write emails
- **sender spool-file**
  - each message to be send is placed (appended) in a designated spool-file by the MUA
- **Mail Transfer Agent (MTA)**
  - program which reads emails from a spool-file in a consuming way
  - forwards these emails into the mailboxes of the recipients (e.g. using SMTP)
- **mailbox**
  - designated file owned by a receiver
  - delivered mails should be appended here

Basic Components



Basic Protocols

- **transport mechanisms to send mails from the sender's spooling memory to the receiver's mailbox:**
  - SMTP - Simple Mail Transfer Protocol (widely used)
  - X.400 (more sophisticated)
- **fetch mechanisms to move (copy) mails from a remote mailbox to a local host**
  - POP - Post Office Protocol
  - IMAP - Internet Message Access Protocol

Basic Protocols

- **multimedia attachment formats:**
  - MIME - Multipurpose Internet Mail Extensions
- **encryption standards:**
  - PGP - Pretty Good Privacy

Email Addresses

- every mailbox can uniquely identified by an email address
- email addresses consists of character strings conforming the following format:

  user@domain

  user: identifies the user or his/her mailbox of a domain

  domain: identifies some organization or a host-machine providing a mail-exchange service (DNS name)

- example: lindner@ict.tuwien.ac.at

Mail Routing in the Internet (DNS based)

- **mail routing service of a mail server can be announced with the help of DNS**
  - DNS servers allow to identify a Mailbox Exchanger (MX) which is registered for a domain
  - using MX-records in the DNS database which specify the name(s) of such machine
    - each MX record is assigned a preference value (positive integer)
    - if several MX server exist for one domain, the MTA will try to transfer the message to the server with the lowest preference value
    - a MTA must not transfer mails to MX servers with a higher preference value than its own (safe way of avoiding mail loops)
  - DNS resolves for any given domain-name the machine's associated IP-address

Message Components (RFC 822, 2822) 1

- **Envelope or Header**
  - contains any information necessary for transmission and delivery
  - starts with a "From" expression in the first line
  - necessary for MUA's mail handling
    - not particular to any transport mechanism (though MTA’s may use some information of the header)
  - contains well defined message information
    - about sender, receiver, intermediate stations, date and time, content-type, return-path (for error messages back to the sender), subject of the message, etc...

Message Components (RFC 822, 2822) 2

- **Body**
  - separated from the header by an empty line
  - contains the user's message
  - maximal 1000 characters
- **Signature**
  - separated from the body by two dashes "--"
  - contains personal information, jokes, PGP-keys or fingerprints, etc.

- **Very important:**
  - Header and Body must be represented with US-ASCII characters only to be RFC822 conform

Header Fields (1)

- **From:**
  - sender's email address and (frequently) her "real name"
  - many formats are used here
- **To:**
  - recipients email address
- **Subject:**
  - what the message is about (to the sender's opinion)
- **Date:**
  - the date the mail *was sent*
- **Reply-To:**
  - hint for the recipient which email address should be used for a reply

Header Fields (2)

- **Organization:**
  - hint which organization (company, etc) the user belongs to
- **Message-ID:**
  - a string, generated by the initial MTA
  - identifies a message uniquely
- **Received:**
  - every site (including sender and recipient) which processes this email inserts such a field in the header
  - several information can be stated here: site name, message-id, time, IP-address, software name
- **X-anything:**
  - used to implement additional features
  - no MUA or MTA should complain about this lines

Simple Mail Transfer Protocol

- **RFC 821, 2821**
- **client-server principle**
  - SMTP relies on TCP, well-known port number 25
- **end-to-end communication**
  - sender (SMTP client) talks directly to the receiver (SMTP server)
  - local deleting condition: mail must successfully arrive at the receiver
- **commands and message-contents are transferred in ASCII format**
  - printable 7-bit US-ASCII (=character values 33-126) plus CR and LF

ASCII-Code

American Standard Code for Information Interchange

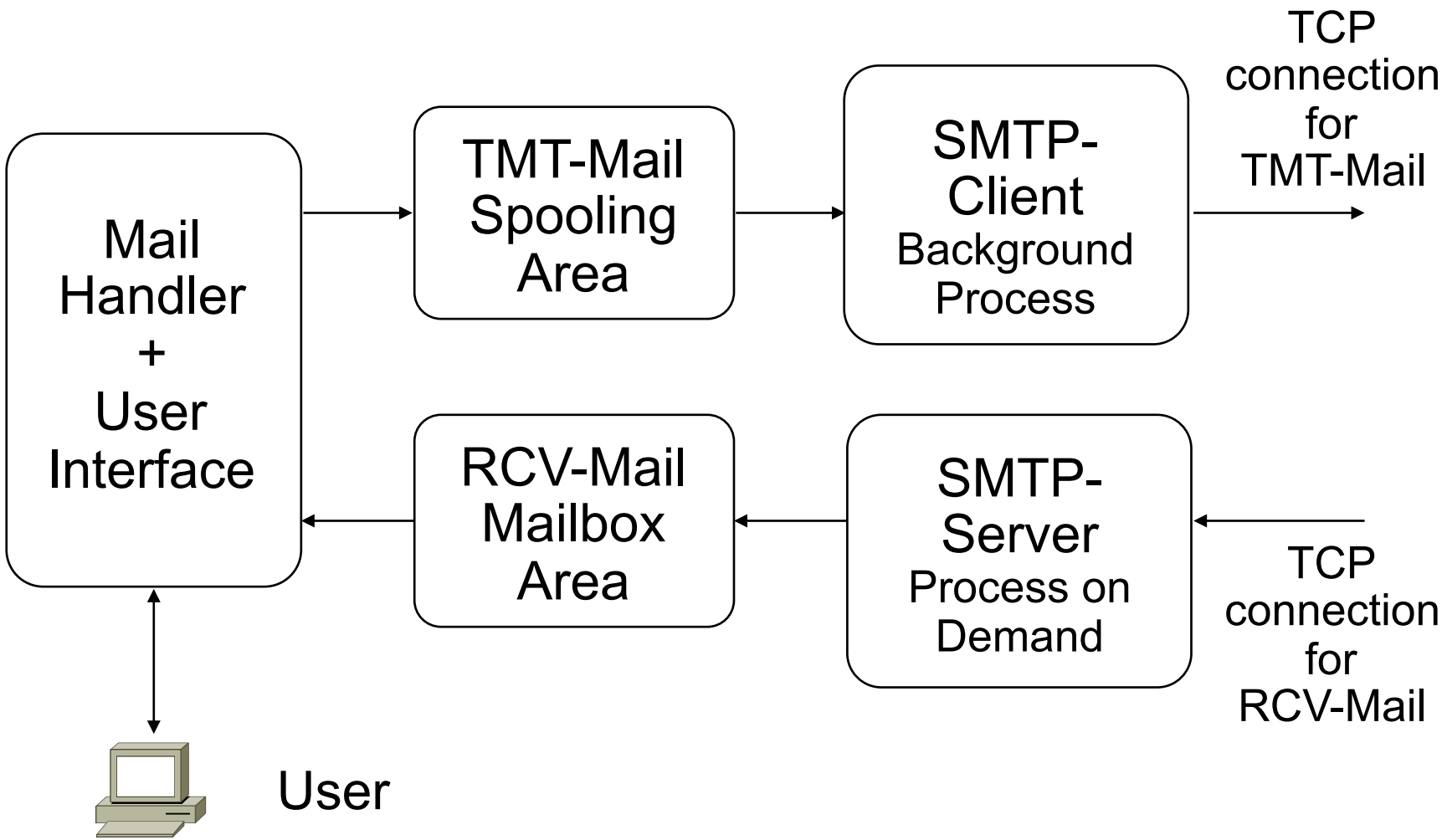
Bit Positions	7	0	0	0	0	1	1	1	1
	6	0	0	1	1	0	0	1	1
	5	0	1	0	1	0	1	0	1
0 0 0 0		Null	DLE	SP	0	@	P	\	p
0 0 0 1		SOH	DC1	!	1	A	Q	a	q
0 0 1 0		STX	DC2	“	2	B	R	b	r
0 0 1 1		ETX	DC3	#	3	C	S	c	s
0 1 0 0		EOT	DC4	$	4	D	T	d	t
0 1 0 1		ENQ	NAK	%	5	E	U	e	u
0 1 1 0		ACK	SYN	&	6	F	V	f	v
0 1 1 1		BEL	ETB	`	7	G	W	g	w
1 0 0 0		BS	CAN	(	8	H	X	h	x
1 0 0 1		HT	EM	)	9	I	Y	i	y
1 0 1 0		LF	SUB	*	:	J	Z	j	z
1 0 1 1		VT	ESC	+	;	K	[	k	{
1 1 0 0		FF	FS	,	<	L	\	l	|
1 1 0 1		CR	GS	-	=	M	]	m	}
1 1 1 0		SO	RS	.	>	N	^	n	~
1 1 1 1		SI	US	/	?	O	_	o	DEL
4 3 2 1									

Transmission Control | Format Control | Printable Character | Information Separator | Others

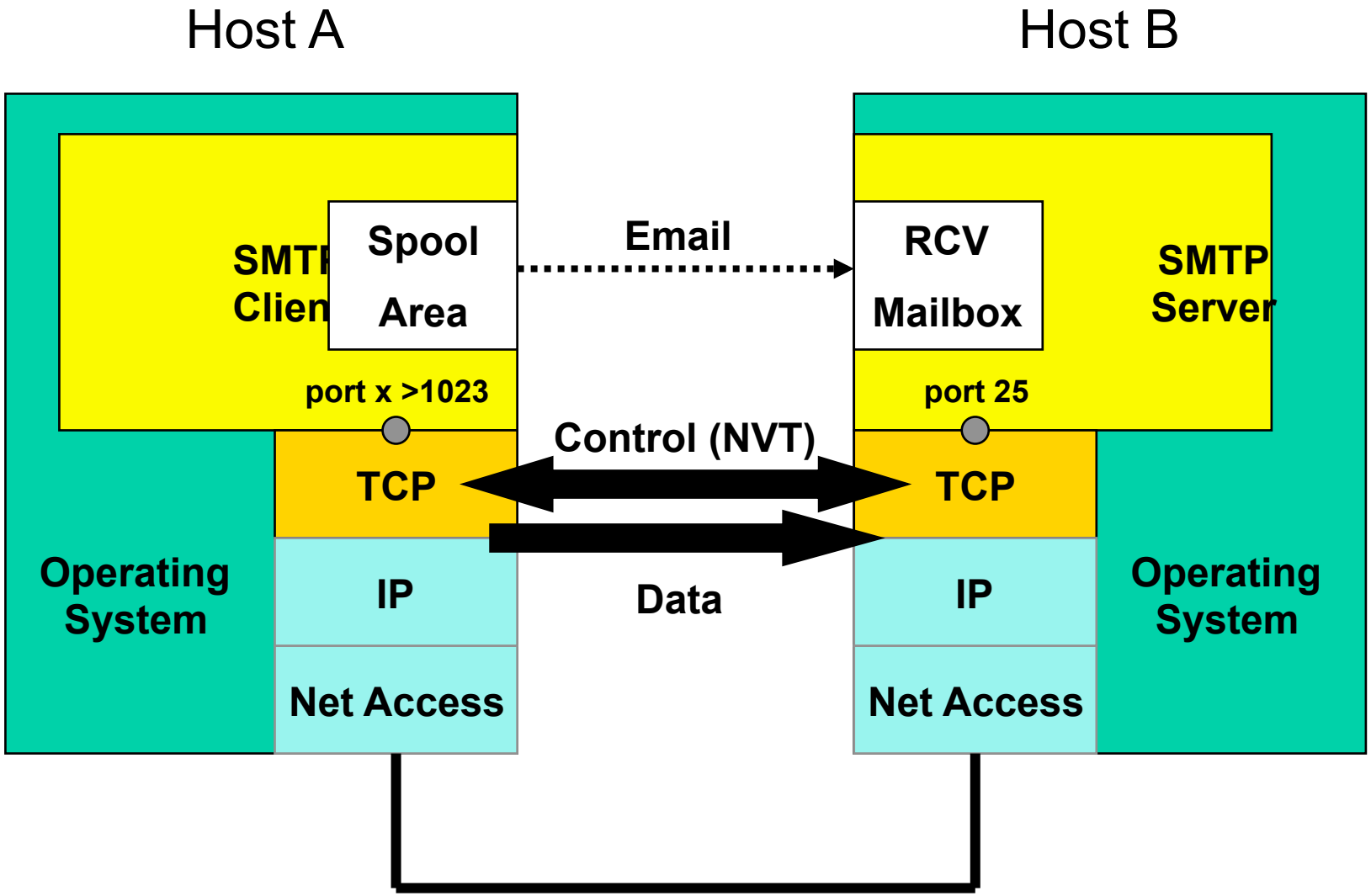
Simple Mail Transfer Protocol - SMTP

- **multiple receivers:**
  - client must establish a separate TCP connection to every receiver's server-process
- **generally, a client background process tries to empty the whole spooling area**
- **mails that cannot be delivered keep waiting in the spooling area**
  - client process will repeat its delivery attempts periodically
  - the user will be noticed about each delivery failure
  - after several repetitions the mail will be removed from the spooling area

SMTP-Model


Mail Handler + User Interface
TMT-Mail Spooling Area
SMTP-Client Background Process
TCP connection for TMT-Mail
RCV-Mail Mailbox Area
SMTP-Server Process on Demand
TCP connection for RCV-Mail
User

SMTP Model


Host A
Host B
SMTP Client
Spool Area
Email
RCV Mailbox
SMTP Server
port x >1023
port 25
Control (NVT)
TCP
TCP
Operating System
IP
Data
IP
Operating System
Net Access
Net Access

SMTP - Commands and Replies

HELO	Authentication
MAIL	Sender‘s Name
RCPT	Receiver‘s Name
DATA	Beginning transmission
SEND	Send directly
SOML	„Send or Mail“
RSET	Reset all Buffers
QUIT	Finnish session

Client → Server
Command

220	Service ready
250	Request mail action ok
354	Start mail input
421	Service not available
450	Request action aborted
500	Syntax Error
550	Requested action not taken
551	User not local
554	Transaction failed

Server → Client
Reply

SMTP-Commands: Client -> Server

- HELO.....for client authentication
- MAIL.....specifies sender's name (FROM-line)
- RCPT.....specifies receiver's name; can be repeated if there are several recipients on the receiver's system
- DATA.....indicates beginning of mail transmission
- SEND..... this email should be send directly to the terminal of the specified user
- SOML..... first act like SEND; if the user's terminal cannot be reached use that user's mailbox ("Send Or MaiL") *
- RSET.....resets all buffers, TCP connection remains open though
- QUIT.....finishes this client-server session

SMTP-Replies: Server -> Client

- 220 <domain> service ready
- 250 <domain> requested mail action okay, completed
- 354 start mail input, end with CR,LF,..,CR,LF
- 421 <domain> service not available, closing trans.cha.
- 450 request action aborted, local error in processing
- 500 syntax error, command unrecognized
- 550 requested action not taken (mailbox not found)
- 551 user not local
- 554 transaction failed

- **error numbers are very similar like those of FTP**
- **both commands and replies are completed with a CR, LF sequence**

SMTP Example (1)

C: (opens TCP connection to port 25 of the server)

S: 220 tuwien.edu Simple Mail Transfer Service ready

C: HELO tugraz.edu

S: 250 OK

C: MAIL FROM: josef@tugraz.edu

S: 250 OK

C: RCPT TO:hans@tuwien.edu

S: 550 no such user there

C: RCPT TO:manfred@tuwien.edu

S: 250 OK

C: DATA

S: 354 start mail input, end with CR LF . CR LF

SMTP Example (2)

C: sends message im RFC 822 Format

```
Date: Sun 17 April 94 09:10:22
From: Josef Maier <josef@tugraz.edu>
Subject: Greetings
To: manfred@tuwien.edu

Did this email reach you?
Josef
```

C: CR , LF , . , CR , LF

S: 250 OK

C: QUIT

S: 221 tuwien.edu closing transmission channel

SMTP Example (3)

Return-Path: josef@tugraz.edu
Posted-Date: Sun 17 April 94 09:10:22 PDT
Received-Date: Sun 17 April 94 09:11:43 PDT
Received: from tugraz.edu by tuwien.edu
id AA07832; Sun 17 April 94 09:11:43 PDT
Date: Sun 17 April 94 09:10:22 PDT
From: Josef Maier <josef@tugraz.edu>
Subject: Greetings
To: manfred@tuwien.edu
(additionally, here may appear some Logging Information caused by SMTP processes having forwarded this mail)

Did this email reach you?
Josef
------

message conforming to the RFC 822 format, seen at the receiver

Post Office Protocol (POP)

- **very often a user reads and writes his emails on a local PC but has his mailbox on a server machine**
  - running a SMTP server process for receiving email
    (probably running also a SMTP client process for sending email)
  - is permanently connected with the Internet
- **POP 3 lets a user fetch his emails from a remote mailbox (client-server principle)**
  - the machine with the mailbox (SMTP-server) runs also a POP3 server process
  - the POP3 client on the user's workstation is able to load and delete emails from that server and also to save them on the local disk

POP3 Principles

- **POP3 relies on TCP**
  - well-known port number 110
  - again commands and error-/state-messages are exchanged using ASCII characters
  - communication procedure is similar to SMTP
- **Some examples of "LAN Mail Access Modules and/or Native Mail Systems"**
  - Pegasus Mail (DOS/Windows)
  - Eudora
  - Groupwise (Novel, IPX based)
  - MS Exchange
  - MS Outlook
  - Lotus Notes

POP3 Commands 1

- USER name … user name for authentication
  - attention: cleartext
- PASS password … password for authentication
  - attention: cleartext
- STAT … to get the number of messages and total size of the messages
- LIST [msg] … if a message number is specified, the size of this mail is listed (if it exists), if not all messages will be listed with the message sizes
- RETR msg .. sends the whole message to the client
- DELE msg … deletes the specified message

POP3 Commands 2

- NOOP … the server does not do anything, just sends a positive response.
- RSET … this command cancels previous delete requests
- QUIT … if entered in the authorization state, it merely ends the TCP connection; if entered in the transaction state, it first updates the mailbox (deletes any messages requested previously) and then ends the TCP connection

Internet Message Access Protocol (IMAP4)

- **RFC 3501**
- **client-server principle**
- **relies on TCP, well-known port 143**
- **IMAP4 is similar to POP3 but more sophisticated**
  - allows a client to access and manipulate emails and mailboxes on a server
  - includes operations for creating, deleting, and renaming mailboxes
  - commands for selective fetching of message attributes
    - ALL
    - BODY
    - BODY<section> (get single pages of a "multipart message"),

IMAP4

- commands for selective fetching of message attributes (cont.)
  - BODYSTRUCTURE (get MIME-1 body structure of a message), ENVELOPE
  - FLAGS (get only the flags that are set for this message)
    - \Seen … Message has been read
    - \Answered … Message has been answered
    - \Flagged … Message is marked for special attention.
    - \Deleted … Message is deleted for later permanent removal.
    - \Draft … Message has been completed.
    - \Recent … Message has arrived recently and this is the first session after its arrival, this flag cannot be changed by the client.
  - FULL
  - RFC822 (get message in RFC822 format)
  - UID (get the unique identifier for this message)

IMAP4

- search-command
  - searches a mailbox for messages that match a given criteria (search keys)
- examine-command:
  - enables *read-only* mailboxes
- maintains several *flags* for each message
  - SEEN, ANSWERED, DRAFT, DELETED, FLAGGED

- **RFC 1733**
  - specifies „Distributed Electronic Mail Models in IMAP4“
    - offline use model
    - online use model
    - disconnected use model

SMTP and Binary Data Sources

- **RFC 822 format**
  - allows only us-ascii characters in the message body
- **For including binary data like pictures, images, executable files in an RFC 822 conform email**
  - they first must be prepared for an ASCII-transmission
    - conversion into 7-bit-Bytes represented by printable ASCII characters
- **several ad hoc methods were used before MIME**
  - UUENCODE and UUDECODE
    - Unix-to-Unix
  - pure hexadecimal representation
  - Andrew Toolkit Representation (ATK)
  - many others

Multipurpose Internet Mail Extensions

- **MIME is a mechanism**
  - for specifying and describing the format of message bodies (content-type) in a standardized way
  - but leaves message body as ASCII text
- **using MIME now emails can contain**
  - images
  - audio-content
  - videos
  - HTML pages
  - application specific data
- **necessary**
  - MUA can identify and support associated content-type

Multipurpose Internet Mail Extensions

- **MIME is realised using**
  - MIME-Version header field
  - Content-Type header field
    - type and subtypes of data in the body
    - this describes how the object within the body is to be interpreted
    - the default value is text/plain; charset=us-ascii,
  - Content-Transfer-Encoding header field
    - this describes how the object within the body was encoded so that it could be included in he message in a mail-safe form (us-ascii-code)
  - Content-Description header field (optional)
    - for additional plain-text data description
  - Content-ID header field
    - a world-unique identifier for the content of this part of the message

7 Standard Content-Types

- **1) text**
  - plain (unformatted text) charset=us-ascii
    - 7 bit (position 0 - 127 in the code table)
  - plain (unformatted text) charset= iso-8859-x (x = 1 - 9)
    - us-ascii plus national characters (position 128 - 255 in the code table)
  - html and enriched
- **2) image**
  - jpeg, gif
- **3) audio**
- **4) video**
  - mpeg

7 Standard Content-Types (cont.)

- **5) application**
  - postscript
  - octet stream
- **6) multipart**
  - mixed:
    - different body parts sequentially presented to the receiver
  - parallel:
    - same as mixed but no order how to presented the different parts to the receiver
  - alternative:
    - different body parts are alternatives of the same information
    - can be presented depending on capabilities of the receiver
    - e.g. email as text/plain or text/html

7 Standard Content-Types (cont.)

- **7) message**
  - the body is an encapsulated message or part of one
  - rfc822
    - encapsulated message is RFC822 conform
  - partial
    - large mail fragmented in smaller pieces
  - external-body
    - pointer to a object existing elsewhere accessible via ftp, tftp, local file, mail-server
- **private types not falling into categories above**
  - starts with a type/subtype X-
    - e.g. X-Mailer (MS Outlook, Novell GroupWise, etc.)
    - e.g. X-Priority (Normal, High, Low)

5 Standard Content-Transfer-Encodings

- **1) 7-bit encoding**
  - body contains strict us-ascii with maximal length of 1000 characters
- **2) 8-bit encoding**
  - possible SMTP agents support the SMTP service extension for 8-bit MIME transport
    - EHLO instead of HELO
  - still maximal length of 1000 characters
- **3) binary encoding**
  - binary with length greater than 1000 characters
  - currently only usable for type=message subtype=external-body

5 Standard Content-Transfer-Encodings

- **4) quoted-printable encoding**
  - real encoding
  - leaves text files largely readable in their encoded form
  - it represents non-mail safe characters by the hexadecimal representation of their ascii-characters
  - non-text characters are replaced by three byte sequence
- **5) Base64 encoding**
  - real encoding
  - for binary data
  - three 8-bit input words -> grouped to 24 bits
  - 24 bits -> grouped to four 6-bit words (bbbbbb)
  - each of it padded to 8-bit (00bbbbbb) word
  - 8-bit word converted with Base64-table to be mail-safe

RFCs

- Mail: RFC 822 (obsolete), RFC 2822
- SMTP: RFC 821 (obsolete), RFC 2821
- POP2: RFC 937
- POP3: RFC 1081, RFC 1225, RFC 1460, RFC 1725, RFC 1939
- POP3 Authentication: RFC 1734
- APOP: RFC 1460, RFC 1725, RFC 1939
- RPOP: RFC 1081, RFC 1225
- IMAP2, IMAP2BIS: RFC 1176, RFC 1732
- IMAP4: RFC 1730, RFC 1731, RFC 1732, RFC 2060, RFC 2061, RFC 3501
- MIME: RFC 2045, 2046, 2047, 2048, 2049

Agenda

- Telnet (Virtual Terminal)
- SSH
- FTP (File Transfer)
- E-Mail and SMTP
- WWW and HTTP

WWW Principles 1

- **Information stored on Web-servers**
  - Documents in HTML format
    - Hypertext Markup Language
  - HTML is a text description language
    - HTML itself is exactly defined by the usage of Standard Generalized Markup Language (SGML)
    - Several HTML versions today
  - SGML is a system for defining structured document types and markup languages to represent instances of those document types
    - HTML is an application of SGML
    - HTML Document Type Definition (DTD) of a document is a formal definition of the HTML syntax in terms of SGML used within this document

WWW Principles 2

- **HTML is a semantic markup language**
  - Within in the text specific “commands” (**Tags**) are included which describes the logical structure of the given text
  - Technically spoken a HTML document consists of elements (containers), which are bracketed by begin- and end-tags
    - **<h[1]>**text-lawa1**</h[1]>** …. for headline
    - ** **text-lawa2** ** ……. for paragraph
    - **<ul** type=„bullettype"**>**
      **
listentry1
**
      **
listentry2
**
      **
** ..... for lists with bullets
    - **bold**
    - *****italic*****

WWW Principles 3

- **Most important element is the link which makes the text “hyper”**
  - text-to-link
  - URL … Uniform Resource Locator -> unique identifier of a given resource in the Internet
    - http://www.ict.tuwien.ac.at/skripten/datenkomm/index.html
- **Tags are device independent**
  - Will be interpreted at the given output system (GUI)
  - GUI … Graphical User Interface
- **WYSIWYM instead of WYSIWYG**
  - What You See Is What You Meant (Get)
  - note: that was the original approach

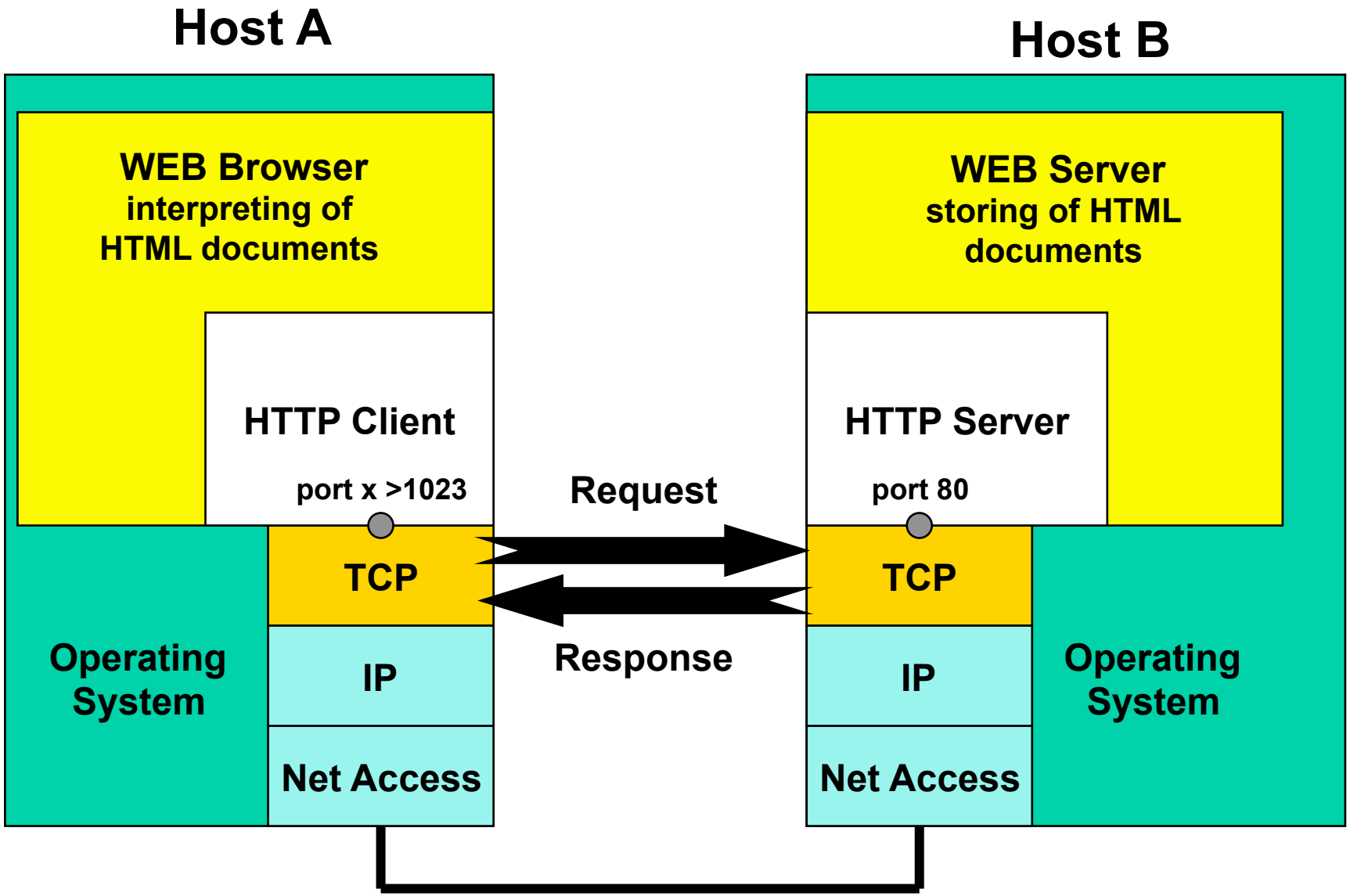
WWW Principles 4

- **Concept of hypertext**
  - Information (text documents) is structured in an hierarchical way. Retrieval of text documents will follow this hierarchy.
  - References (“links”) within an document allow access to other documents located at a different level of hierarchy
  - Allows a new way of navigating within text documents
  - Allows links to documents residing on other machines
  - Later expanded to hypermedia
    - Including graphics, audio and video
- **Hypertext and GUI (Browser)**
  - The base for the success of the World-Wide-Web

WWW Principles 5

- **Web Browser**
  - Can download and present HTML documents to the user
  - E.g. NCSA Mosaic, Netscape Navigator, Microsoft Internet Explorer, Opera
- **Web Browser use HTML Interpreter for the presentation of documents**
- **Web Browser use HTTP protocol for the download of documents**
  - Hypertext Transfer Protocol
  - Client - Server based
    - Browser as client, WEB Server - as server
  - Server accessible via well-known TCP port 80

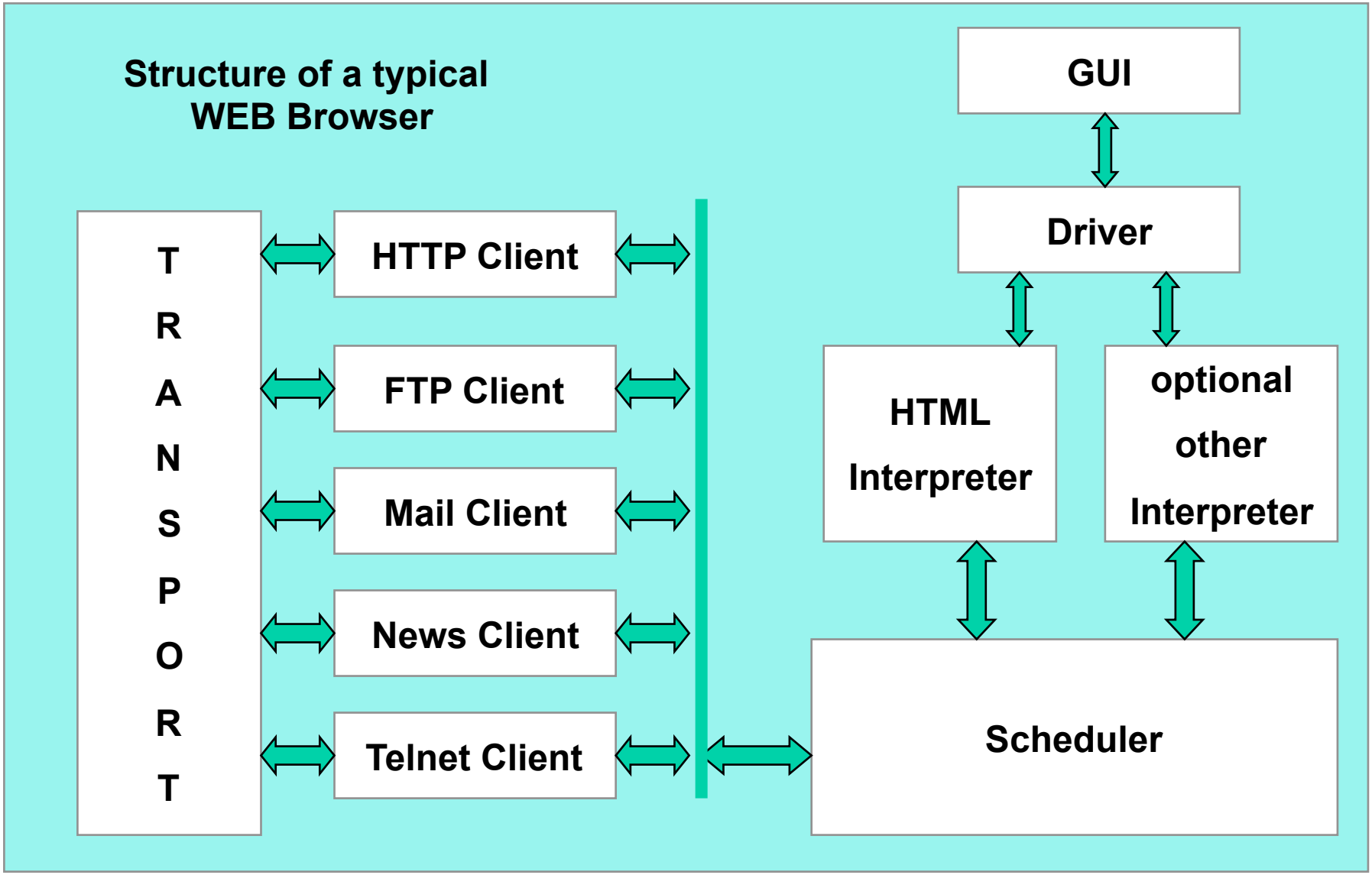
HTTP - WWW Model



WWW Principles 6

- **Web Browsing**
  - Usage of cache techniques to reduce network load and improve performance
    - Browser Cache
    - Proxy Server as caching system
    - Document attributes for decision what is newer or must be refreshed
      - Time and Date
      - Meta Information http-equiv “expires”
  - Good for static Web content
  - More complicated for dynamic Web content

WWW Principles



Introduction to HTML Tutorial

- **SELFHTML by Stefan Münz**
- **Excellent InfoBase about HTML**
- **Portal:**
  - http://selfaktuell.teamone.de/
  - http://selfhtml.teamone.de/
  - http://aktuell.de.selfhtml.org/
- **Download of selfhtml80.zip (7 MB) allows you to install tutorial on your PC**

HTTP Principles

- **Hypertext Transfer Protocol**
  - First Version 1.0 (RFC 1945)
  - Current Version 1.1 (RFC 2616, 2817)
- **Base for transport of WWW documents**
  - Between client (Browser) and server (Web-Server)
- **On top of TCP**
  - Hence connection-oriented
  - Well-know server port 80
- **Stateless**
  - Client opens a TCP connection, requests a document, server responds with document, client closes TCP connection (remedy ->cookies)

HTTP Characteristics

- **Application-level protocol**
  - With the lightness and speed necessary for distributed, collaborative, hypermedia information systems
- **Object-oriented protocol**
  - Methods are applied on objects (sequentially)
- **HTTP messages consist of**
  - Header
  - Body
- **HTTP allows usage of a set of methods**
  - Methods specify the purpose of a request
  - Methods are applied on URLs included in the header

HTTP Header and Body

- **Header**
  - contains the URL, method, and parameters
  - HTTP v1.0 methods: GET, HEAD, POST
- **Body**
  - Contains user data described by a MIME header
    - MIME = "Multipurpose Internet Mail Extensions"
    - Also used by Internet Mail
  - User data can be
    - HTML information (= a web page)
    - Graphics, videos, sound-data, ...

HTTP Messages

- A client establishes a connection with a server and sends a request to the server containing
  - a request method
  - URL
  - protocol version
  - MIME-encoded data (optionally)
- The server's response contains
  - a status line
    - containing messages protocol version
    - and a success or error code
  - MIME-encoded data
    - Server information (e.g. expiration time for cache)
    - entity metainformation
    - and possible body content

HTTP v1.0 Methods

- **GET**
  - This method allows the client to retrieve the data which was determined by the request URL.
- **HEAD**
  - This method allows the client to retrieve meta-information about the entity which does not require to transfer the entity body
    - Check if document was changed since last retrieval and hence to be refreshed
- **POST**
  - This method allows the client to store documents on the server. The post function may be supported by the server.

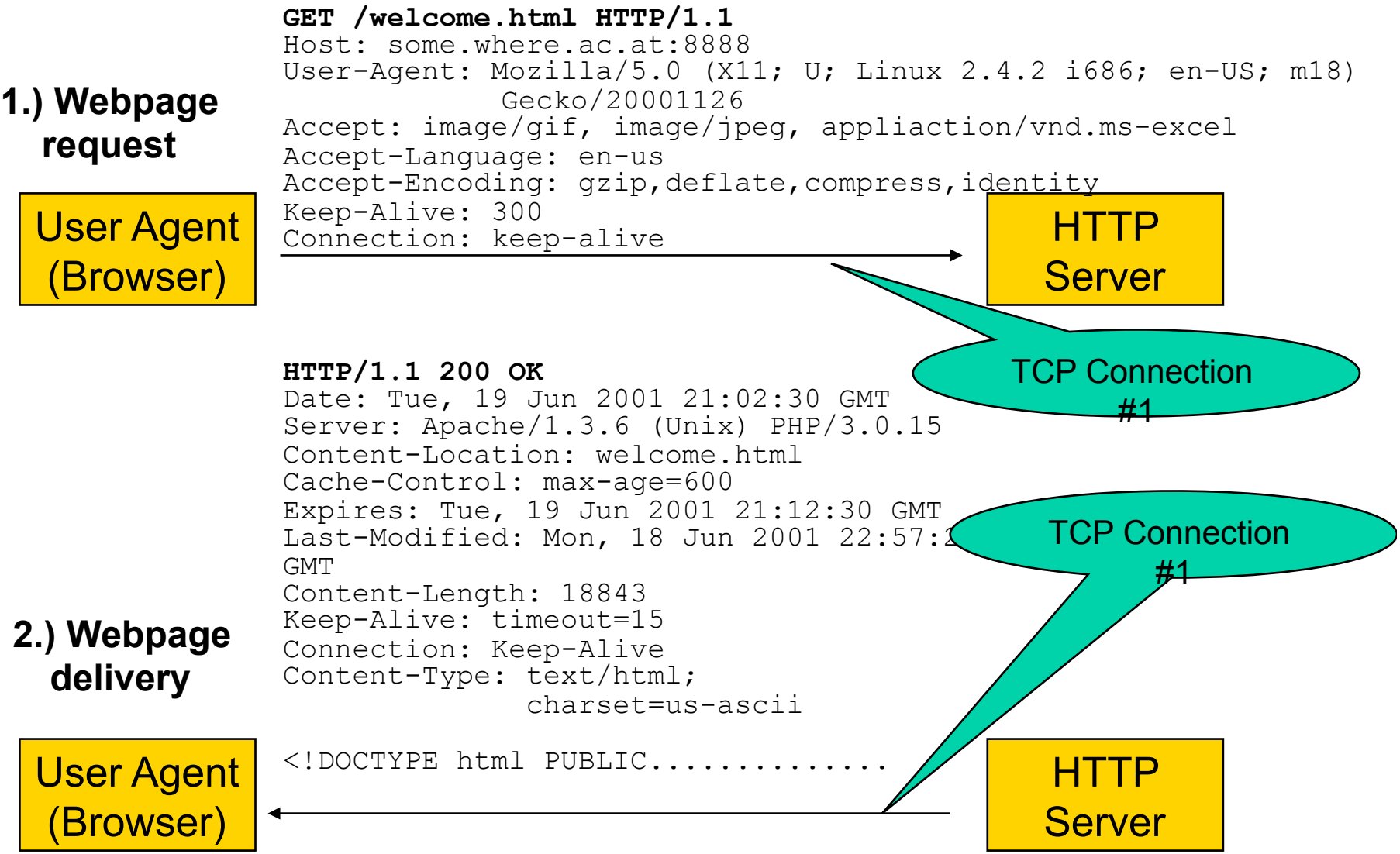
HTTP v1.1 Methods

- **PUT**
  - This method is similar to the post method with one important difference which is the URL in post request identifies the resource that will handle enclosed entity whereas with put request the URL identifies the enclosed entity itself.
- **DELETE**
  - This methods requests that the server delete the source determined by the request URL.
- **TRACE**
  - Trace method allows the client to see how the message was retrieved at the other side for testing and diagnostic purposes (remote application-layer loopback)

HTTP Request Methods and CGI data

- **HEAD or GET request**
  - Only headers, no body
  - Form data for CGI is encoded in HTTP_QUERY_STRING
    - CGI script receives the environment variable QUERY_STRING which contains the whole information
- **POST request**
  - Header and body
  - Body contains user data (also form data)
- **Other differences**
  - HEAD request does not expect body in the response message
  - GET and POST accept responses with or without body

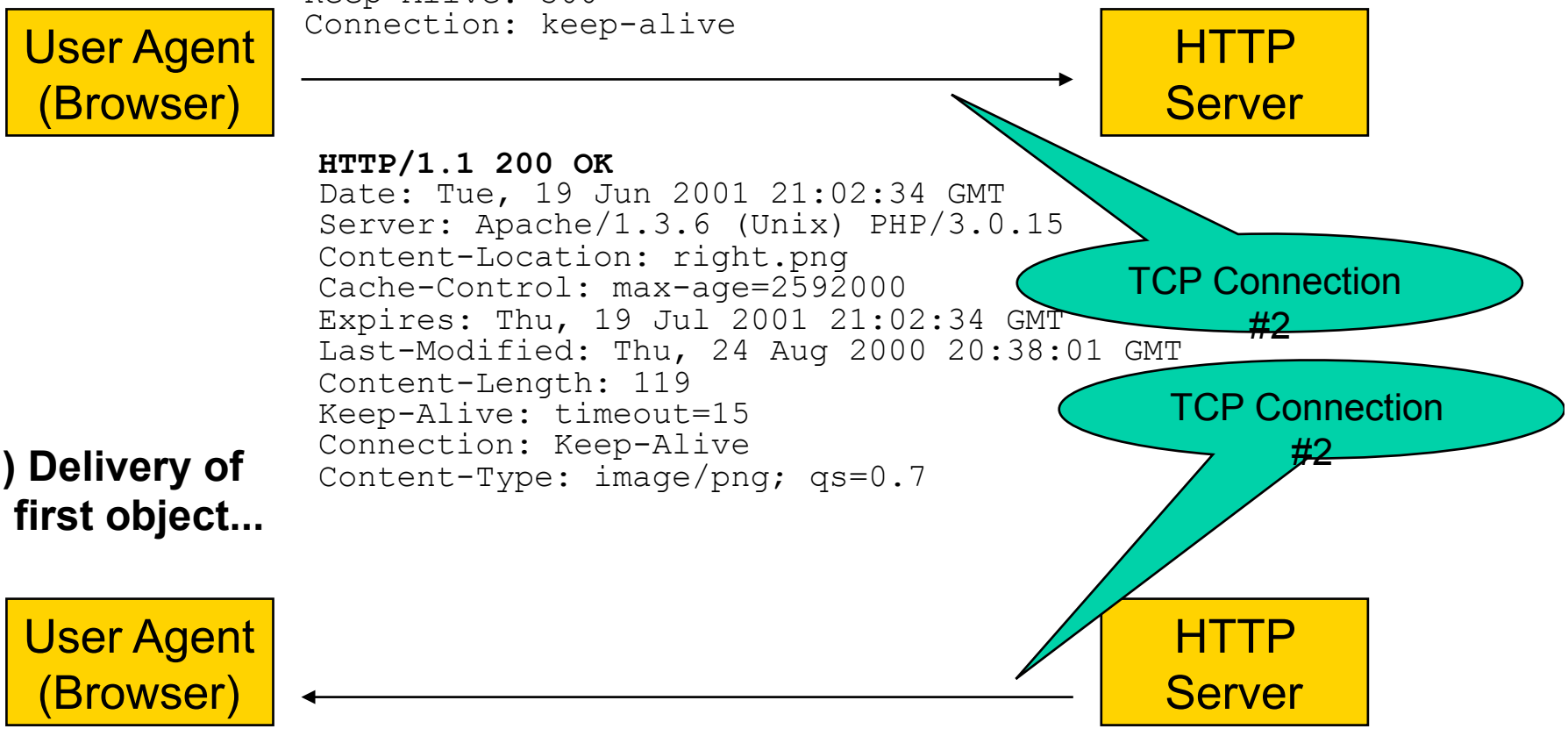
HTTP Communications (1)



HTTP Communications (2)

3.) Request for contained objects (icons,...)

```
GET /Icons/right HTTP/1.1
Host: some.where.ac.at:8888
User-Agent: Mozilla/5.0 (X11; U; Linux 2.4.2 i686; en-US; m18)
            Gecko/20001126
Accept: image/gif, image/jpeg, appliaction/vnd.ms-excel
Accept-Language: en
Accept-Encoding: gzip,deflate,compress,identity
Keep-Alive: 300
Connection: keep-alive
```




```
HTTP/1.1 200 OK
Date: Tue, 19 Jun 2001 21:02:34 GMT
Server: Apache/1.3.6 (Unix) PHP/3.0.15
Content-Location: right.png
Cache-Control: max-age=2592000
Expires: Thu, 19 Jul 2001 21:02:34 GMT
Last-Modified: Thu, 24 Aug 2000 20:38:01 GMT
Content-Length: 119
Keep-Alive: timeout=15
Connection: Keep-Alive
Content-Type: image/png; qs=0.7
```

4.) Delivery of first object...

HTTP Communications (3)

- **Current HTTP communications requires**
  - That the connection is established by the client prior to each request
  - And closed by the server after sending the response

- **If a webpage consists of several components**
  - Every component is downloaded by a separate TCP connection even if the component resides on the same machine !!!!
  - In HTTP v.1.1 concept of persistent connections
    - Objects of same type are retrieved via single TCP connection

HTTP 1.1 (RFC 2616)

- **HTTP/1.1 was developed**
  - to overcome version 1.0 problems
    - HTTP/1.0 does not sufficiently take into consideration the effects of hierarchical proxies, caching, the need for persistent connections, and virtual hosts
    - GET, HEAD or POST method only
    - Basic Authentication (Base64 coding of user-id, passw.)
  - to make HTTP a good Internet citizen
    - in HTTP/1.1, a persistent connection may be used for one or more request/response exchanges
  - to increase functionality
    - GET, HEAD, POST, PUT, DELETE, TRACE methods
    - caching control support to improve overall performance

HTTP 1.1 Associated Documents

- **RFC 2617 Basic and Digest Authentication**
  - Eliminates clear-text password of basic authentication
  - Uses cryptographic hashes (MD5)
  - Still the problem remains how to distribute a common secret safely between agent and server
- **RFC 2964, 2965 State Management Mechanism**
  - Specifies a way to create a stateful session with HTTP requests and responses
  - Based on two new headers
    - Cookie and Set-Cookie which carry state information between participating origin servers and user agents

Cookies

- **Originally, developed by Netscape Corporation**
- **To circumvent the stateless nature of HTTP**
  - HTTP servers respond to each client request without relating that request to previous or subsequent requests
  - Difficult to create services such as virtual shopping carts
- **"Cookies" introduce session information**
- **Two additional HTTP headers**
  - Set-Cookie
  - Cookie

What is a Cookie?

- **A Cookie**
  - Small piece of information (a string)
  - Basically, a special HTTP header (sent by the server)
  - Returned by the browser for each reconnection
- **String contains up to five attributes**
  - Name and value
  - Domain
  - Path
  - Lifetime
  - Security Info

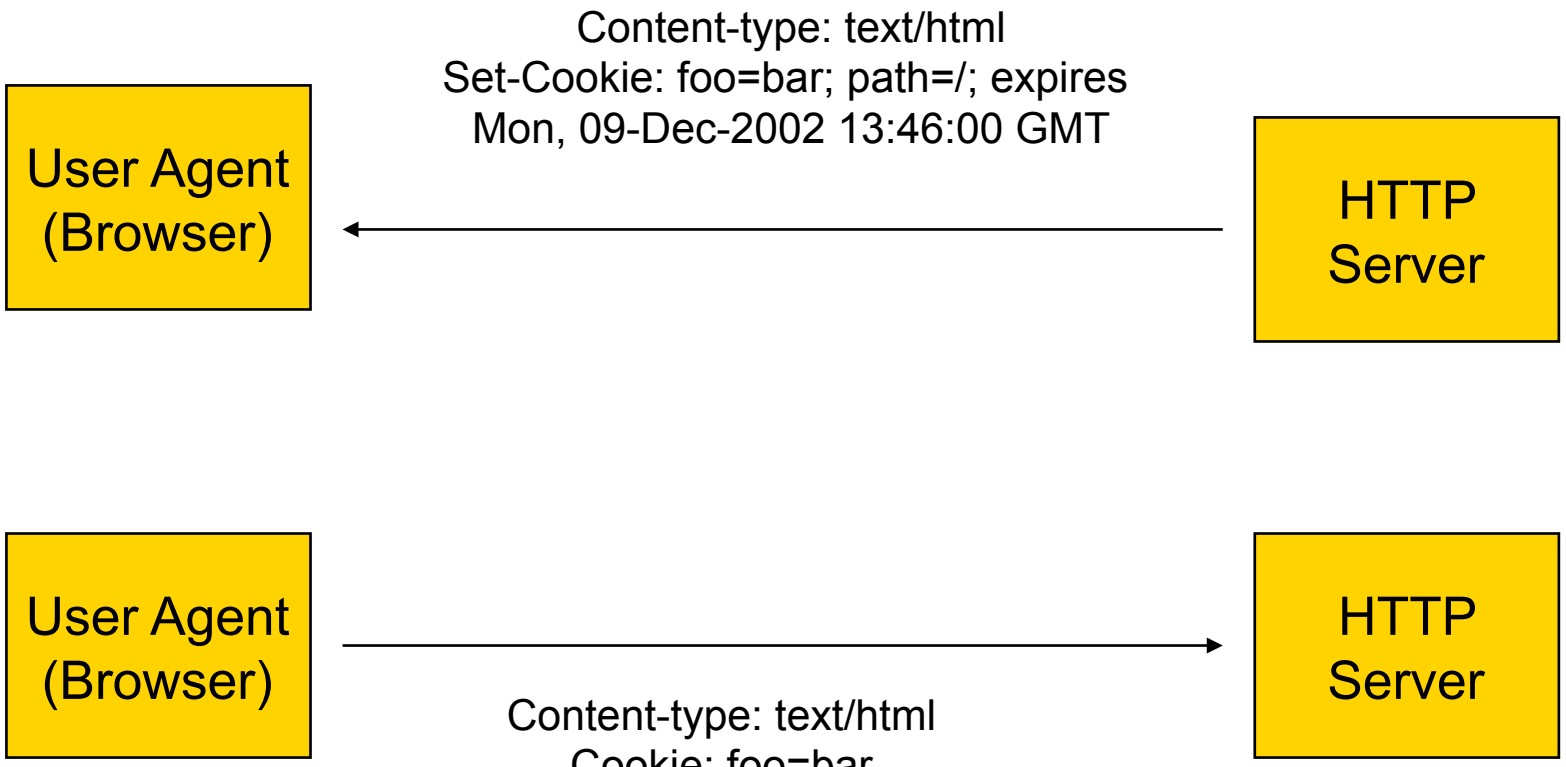
Cookie Structure

- **Name and value**
  - The actual information (<name> = <value>)
  - Usually a session ID
  - Only these two parameters are mandatory!
- **Domain**
  - The domain the cookie is valid for
  - Tells browser about valid domain names whose servers would recognize this cookie
  - Promiscuous cookies are not allowed
  - Domain attribute must not contain top level domains

Cookie Structure

- **Path**
  - Scope of validity on the server's site
  - Pages outside of that path cannot read or use the cookie
  - E.g. "/ " means: valid for the entire site
- **Lifetime**
  - = "Expiration date"
  - If the lifetime is longer than the time the user spends at that site, then this cookie is saved on disk for future reference
- **Security Info**
  - Whether a secure connection must be established before sending this cookie (E.g. SSL)

HTTP Cookie Header Exchange



Cookie Risks

- **Note: Often used to track user behaviour !**
- **Advertisers smuggle cookies on your disk!**
  - Since most banners are references to other sites
    - E.g. "DoubleClick" (ad.doubleclick.net)
  - One of the most controversial issues of the Web!

- **Cookies can be turned off on most browsers**
  - But some sites might not continue its service
  - Possible remedy for Unix machines
    - Create a symbolic link from the cookies directory to /dev/null

Web-Browsers

- **Browsers**
  - Also known as "User Agents"
  - Use HTTP to access special encoded documents from a web-server
    - This documents are usually encoded in HTML
  - Should correctly display document content
- **Very complex applications**
  - Must handle different HTML versions
  - Cascading Style Sheets
  - JavaScript
  - Several graphic formats
  - Security options
  - Several additional services
    - FTP, MAIL, and USENET capabilities

Web-Browsers

- **Big browser companies intentionally write non-standard-conforming Browser software**
- **Who controls Web-standards?**
  - Led to "Browser-Wars"
  - Netscape vs. MS Internet Explorer
- **Other Browsers**
  - Opera
  - Lynx
    - Very fast text browser
  - Mozilla
    - Best implementation of W3C HTML standards
    - Most sophisticated HTML engine "Gecko"
    - OpenSource
  - etc.

Web-Browsers Dynamic 1

- **So far we have handled a static behaviour only**
  - The client requests a document and the server provides the content

- **To make the behavior more dynamic on the client side**
  - JavaScript / JScript
  - JavaApplets
  - ActiveX
  - Flash

Web-Browsers Dynamic 2

- **JavaScript (ECMA-Script)**
  - HTML extension and programming language by NetScape
  - Now standardized by ECMA-262
  - JavaScript programs are embedded as a source directly in an HTML document
    - Structures like frame, form, window are implemented
  - The program is executed on the client browser while the downloaded HTML is interpreted
    - Browser must be JavaScript enabled
  - JavaScript programs can control the behaviour of forms, buttons and text elements. In addition, they can be used to create forms whose fields have built-in error checking routines.
  - JScript is Microsoft´s answer to JavaScript

Web-Browsers Dynamic 3

- **JavaApplets**
  - Java is a platform-independent, object-oriented programming language inspired by C and C++ developed by SUN
  - As part of an HTML document Java programs are downloaded from the server and executed on the client within a restricted area (Java Virtual Machine (JVM) -> “sandbox” ; execution by interpreter),
  - A Java program started from inside an HTML (Web) page is called a Java Applet as opposed to a Java program, which is executed from the command line or otherwise on the local system
  - Java Applets were not supposed to touch anything local (outside of its JVM), and could only communicate back to theserver it was downloaded from.
  - With Java 1.1, applets can be signed with security keys and certificates and can therefore be authenticated. Thus, an applet can be authorized to access local resources, such as file systems, and it may communicate with other systems.

Web-Browsers Dynamic 4

- **ActiveX**
  - Is Microsoft´s answer to Java
  - Could be used to make MS-OS specific things visible and usable for the WEB
    - e.g. content from HTML via OLE to Excel-table
  - ActiveX Controls could be compared with Java Applets but there is no “sandbox” principle
    - User can specify barrier of trust only (similar to Internet Explorer)
  - Compiler must support **Component Object Model** (COM)
- **Flash**
  - Is a proprietary SW product which can be used for animation of Web-pages
  - It works just like a plugin by opening the corresponding program
- **Both ActiveX and Flash are not Internet standards**

Web-Servers

- **Web-servers are basically http-servers**
  - Listen at port 80
- **Examples**
  - httpd NCSA
    - First http server
  - Apache
    - Most frequently used
    - Freeware
  - Internet Information Server (IIS)
    - Microsoft
  - Netscape Communications Server
- **Targets for DoS Attacks**
  - No solution against it!

Web-Servers Dynamic 1

- **So far we have handled a static behaviour only**
  - The client requests a document and the server provides the content

- **To make the behaviour more dynamic on the server side**
  - Common Gateway Interface (CGI)
  - PHP (Hypertext Preprocessor)
  - Active Server Pages
  - Servlets
  - Server-Sides-Include (SSI)
  - Java Server Pages (JSP)

Web-Servers Dynamic 2

- Common Gateway Interface (CGI)
  - Allows a Web server to execute a program that is provided by the Web server administrator, rather than retrieving a file.
  - CGI programs allow a Web server to generate a dynamic response, based on the client's input.
  - A variety of programming languages (C, Pascal, etc) can be used to develop programs that interface with CGI. In principle any compiled code could be executed, but normally only Scripts are used which are interpreted at the running time.
  - The most popular interpreter is PERL (Practical Extraction and Report Language) because programs are easily portable across platforms.

Web-Servers Dynamic 3

- PHP (Hypertext Preprocessor)
  - Alternative to CGI/PERL
  - PERL not optimized for dynamic Web pages
  - PHP is such an optimization
  - HTML documents stored on Web-servers can contain PHP programs.
  - If a client requests such a HTML document the server executes this program (interpreter), generates the final HTML-Code and transport the requested HTML document to the client
- Active Server Pages
  - Microsoft´s answer to PHP

Web-Servers Dynamic 4

- Servlets
  - In order to spare resources on clients and networks, Java Applets can be executed on the server rather than downloaded and started at the client. Such programs are then referred to as Servlets.
  - Servlets are Java Applets running at the server side
- Server-Sides-Include (SSI)
  - SSI is a technology which allows a Java enabled Web-server to convert a section of an HTML file into an alternative dynamic portion each time the document is sent to the client's browser.
  - This dynamic portion invokes an appropriate Servlet and passes to it the parameters it needs. Servlets may not be written in Java.
  - The replacement is performed at the server and it is completely transparent to the client.
  - Pages that use this technology have the extension .shtml instead of .html (or .htm).

Web-Servers Dynamic 5

- Java Server Pages (JSP)
  - This is an easy-to-use solution for generating HTML (or other markup languages such as XML) pages with dynamic content.
  - A JSP file contains combinations of HTML tags, NCSA tags (special tags that were the first method of implementing server-side includes), <SERVLET> tags, and JSP syntax.
  - JSP files have the extension .jsp.
  - JSP can be used to access reusable components, such as Servlets, JavaBeans (reusable Java objects), and Java-based Web applications. JSP also supports embedding inline Java code within Web pages.