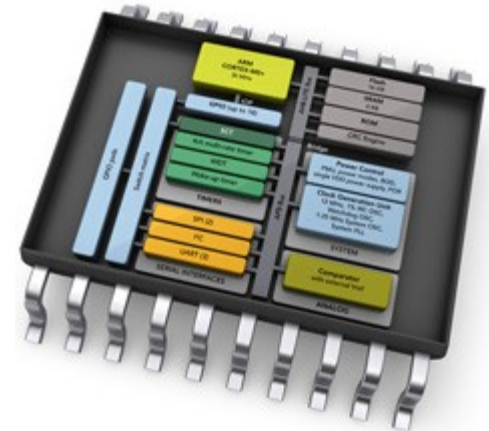


Kommunikationssysteme

(Modulcode 941306)

Prof. Dr. Andreas Terstegge



Kurzer Überblick über:

- HTTP
- SSL / TLS
- FTP / SFTP
- Email: SMTP / POP3 / IMAP
- Telnet / ssh
- SNMP

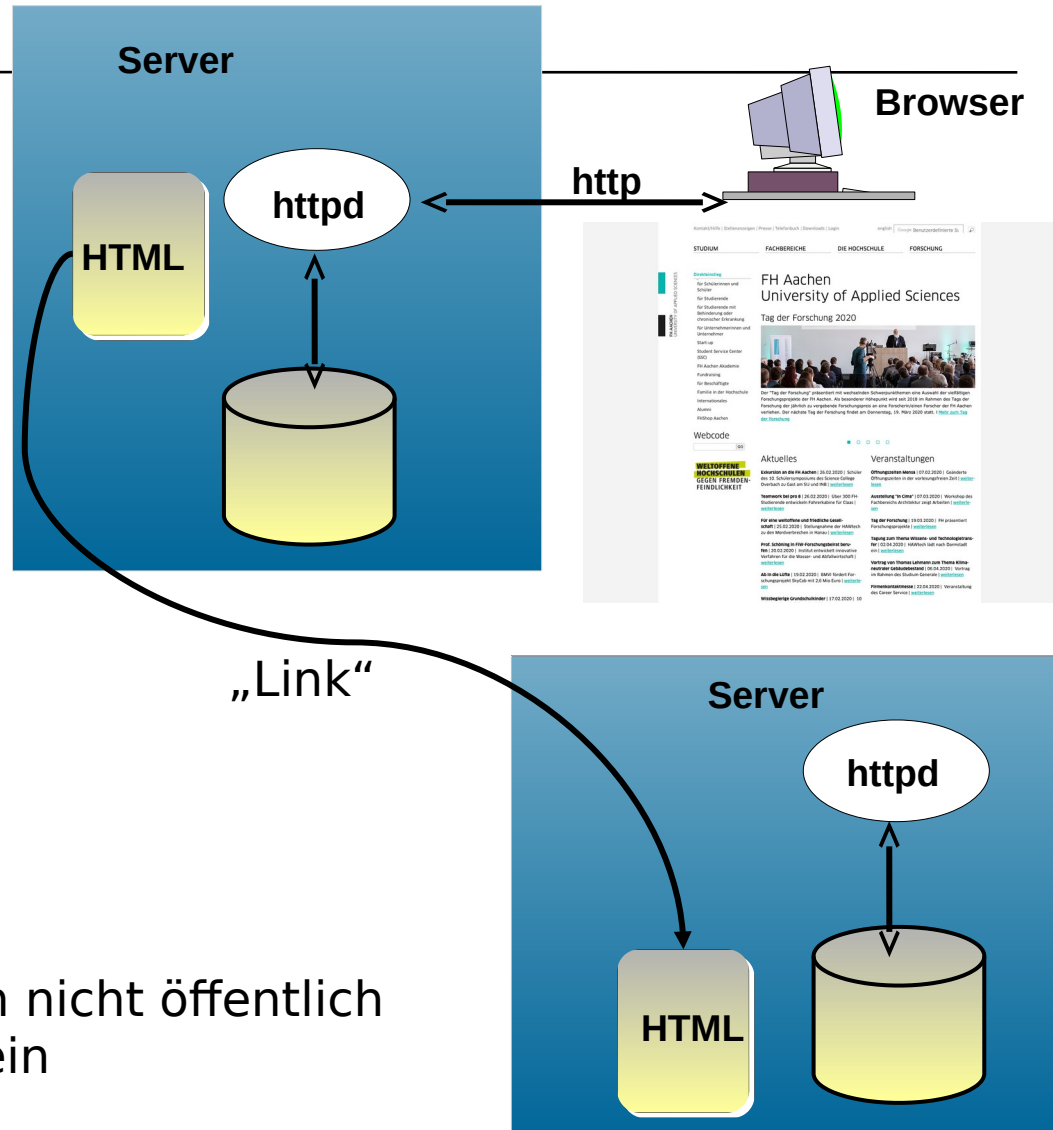
Anwendungsprotokolle

```
tcpmux      1/tcp          # TCP port service multiplexer
echo        7/tcp
echo        7/udp
discard     9/tcp          sink null
discard     9/udp          sink null
systat      11/tcp         users
daytime     13/tcp
daytime     13/udp
netstat     15/tcp
gotd        17/tcp         quote
chargen     19/tcp         ttytst source
chargen     19/udp         ttytst source
ftp-data    20/tcp
ftp         21/tcp
fsp         21/udp         fspd
ssh         22/tcp         # SSH Remote Login Protocol
telnet      23/tcp
smtp        25/tcp         mail
time        37/tcp         timserver
time        37/udp         timserver
whois       43/tcp         nicname
tacacs      49/tcp         # Login Host Protocol (TACACS)
tacacs      49/udp
domain      53/tcp         # Domain Name Server
domain      53/udp
bootps      67/udp
bootpc      68/udp
tftp        69/udp
gopher      70/tcp         # Internet Gopher
finger      79/tcp
http        80/tcp         www          # WorldWideWeb HTTP
kerberos    88/tcp         kerberos5 krb5 kerberos-sec # Kerberos v5
...
```

Linux: Zuordnung
der Portnummern in
/etc/services

Web-Anwendungen: HTTP

- Internet-basierte Client/Server-Architektur
 - Browser (Client) zur graphischen Darstellung
 - HTTP-Server zur Übertragung der Daten und Dokumente
- **HTML**: Sprache zur Beschreibung der Seiten
- **HTTP**: Protokoll zur Übertragung der Seiten
- **TCP**: Von HTTP verwendetes Transportprotokoll
- **URL**: Spezifikation von Ort und Zugriffsmodalitäten



Web-Anwendungen müssen nicht öffentlich zugänglich sein

URI (Uniform Resource Identifier)

URI dient der eindeutigen Adressierung von abstrakten und physikalischen Ressourcen im Internet (Spezifikation in RFC2396)

URI: URLs u URNs

URI

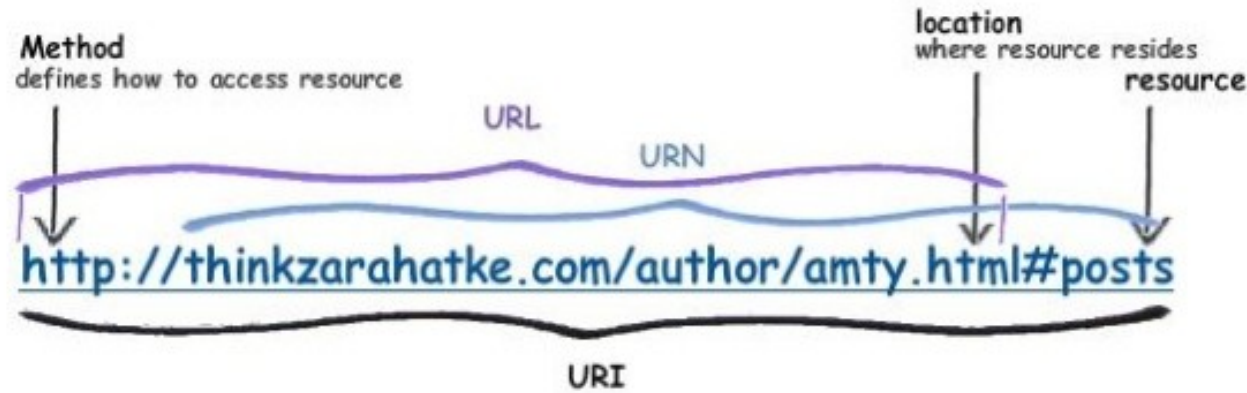
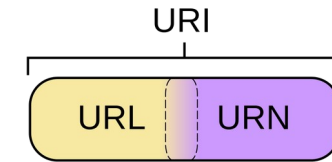
URL (Uniform Resource Locator)

Adressierung von Informationsobjekten mit Festlegung des Zugangs-Protokolls (Ort der Ressource). RFC2141

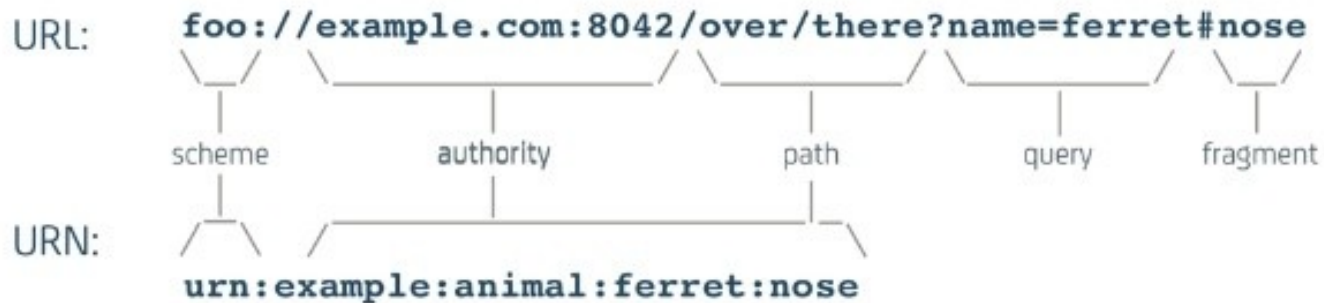
URN (Uniform Resource Name)

Adressierung von Objekten ohne ein Protokoll festzulegen (Eindeutige und gleichbleibende Referenz – Name der Ressource). RFC1738

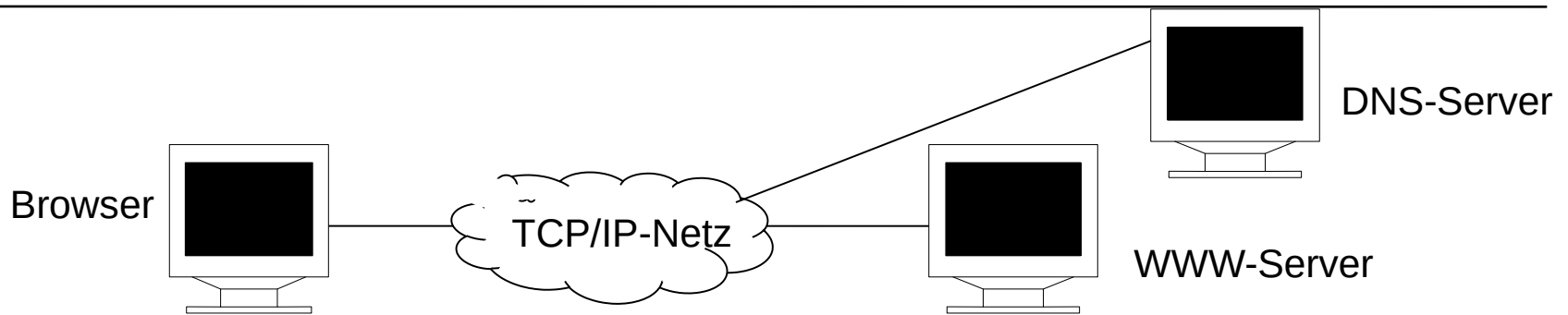
URI (Uniform Resource Identifier)



The structure of URIs



Abruf von Webseiten



Browser fragt DNS nach der IP-Adresse des Servers



DNS antwortet



Browser öffnet eine TCP-Verbindung zu Port 80 des Rechners



Browser sendet das Kommando GET /material/allgemein.html



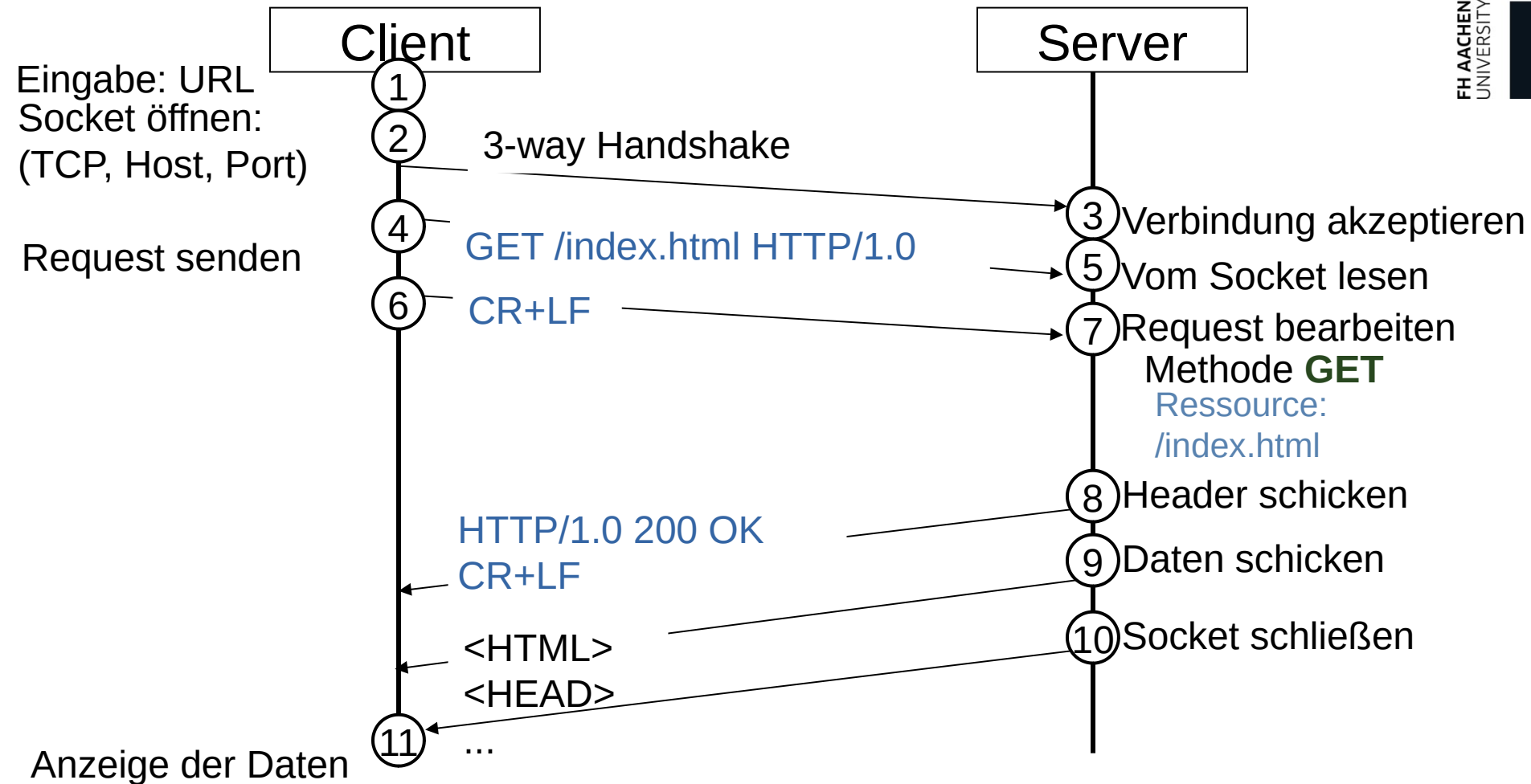
WWW-Server schickt die Datei allgemein.html zurück



Verbindung wird wieder abgebaut.



Der Ablauf einer HTTP-GET-Anfrage



HTTP Request Header

method	sp	URL	sp	version	cr	lf
header field name			:	value	cr	lf
header field name			:	value	cr	lf
⋮						
header field name			:	value	cr	lf
cr	lf					
data						

Request line: notwendiger Teil, z.B.

GET server.name/path/file.type

Header lines: optional, weitere Angaben zum Host/Dokument, z.B.

Accept-language: fr

Entity Body: optional. Weitere Angaben, falls der Client Daten überträgt (*POST-Method*)

Request-Beispiel (GET):

GET /index.html HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/27.0.1453.116 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: de-DE, de;q=0.8, en-US;q=0.6, en;q=0.4

Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1

Accept-Encoding: gzip, deflate, sdch

Connection: keep-alive

HTTP Response Header

version	sp	status code	sp	phrase	cr	lf
header field name		:	value	cr	lf	
header field name		:	value	cr	lf	
⋮						
header field name		:	value	cr	lf	
cr	lf					
data						

Entity Body: erfragte Daten

Status line: *status code* und *phrase* übertragen das Ergebnis einer Anfrage und eine zugehörige Meldung, z.B.

- 200 OK**
- 400 Bad Request**
- 404 Not Found**

Gruppen von Status-Meldungen:

- 1xx: Nur zur Information
- 2xx: Erfolgreiche Anfrage
- 3xx: Redirection, es müssen weitere erforderliche Aktivitäten durchgeführt werden
- 4xx: Client-Fehler (Syntax)
- 5xx: Server-Fehler

Response-Beispiel:

HTTP/1.1 200 OK

Date: Wed, 26 Jun 2013 16:36:27 GMT

Server: Apache

Content-Type: text/html; charset=UTF-8

Content-Length: 12313

Last-Modified: Mon, 16 Apr 2013 20:27:06 UTC

Connection: keep-alive

<!DOCTYPE html>

<html>

...

HTTP Anfragen/Methoden

GET	retrieve information
HEAD	retrieve resource headers
POST	submit data to the server.
PUT	save an object at the location
DELETE	delete the object at the location

HTTP

Binärdaten und das textbasiert

HTTP ist textbasiert (eMail auch!)

Wie können binäre Daten übertragen werden?

- Die Antworten des Servers auf eine vollständige GET-Request beinhaltet MIME-Informationen
- **MIME = Multipurpose Internet Mail Extensions**
- Definiert die Kodierungsregeln für Nicht-ASCII-Nachrichten
- MIME ermöglicht die Nutzung verschiedener Kodierungen (media types) in einer Nachricht

Die “Content-Type:”-Zeile im MIME-Header legt den Datentyp (type/subtype) einer Nachricht fest

Content-Transfer-Encoding: definiert die Transfersyntax, in der die Daten des Hauptteils übertragen werden, wird aber bei HTTP nicht benutzt

- Content-Encoding und Transfer-Encoding Felder

Beispiele:

- Content-Type: text/html
- Content-Type: image/GIF

HTTP

Binärdaten und das textbasiert

MIME-Version: 1.0
Content-Type: MULTIPART/MIXED;
BOUNDARY= "8323328-2120168431-824156555=:325"
--8323328-2120168431-824156555=:325
Content-Type: TEXT/PLAIN; charset=US-ASCII
A picture is in the appendix
--8323328-2120168431-824156555=:325
Content-Type: **IMAGE/JPEG**; name="picture.jpg"
Content-Transfer-Encoding: **BASE64**
Content-ID: <PINE.LNX.3.91.960212212235.325B@localhost>

/9j/4AAQSkZJRgABAQEAlgCWAAD/
2wBDAAEBAQEBAQEBAQEBAQEBAQIBAQEBAQIBAQECAGICAGICAGIDAwQDAwMDA
wICAwQDAwQEBAQEAgMFBQQEBQQEBAT/2wBDAQEBAQEBAQIBAQIEAwIDBAQEBA
[...]
KKACiigAooooAKKKKACiigAooooAKKKKACiigAooooAKKKKACiigAoooo
AKKKKACiigAooooAKKKKACiigAooooAKKKKACiigAooooAKKKKACiigAo
oooAD//Z
---8323328-2120168431-824156555=:325 --

Virtual Hosts

- Auf einem Rechner sollen verschiedene Domains und Web-Server zur Verfügung stehen
→ Jeder Server hat die gleiche IP, aber ggf. unterschiedliche DNS-Namen!
- Ein oder mehrere Webserver (Software) sollen die Anfragen, für die auf dem Rechner vorhandenen Domains, beantworten
- Typische Anwendung: Web-Hosting (Provider)

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
          (KHTML, like Gecko) Chrome/27.0.1453.116 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: de-DE, de;q=0.8, en-US;q=0.6, en;q=0.4
Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1
Accept-Encoding: gzip, deflate, sdch
Connection: keep-alive
```


HTTP Evolution

Standard:	RFC 1945 🔗 HTTP/1.0 (1996)
	RFC 2616 🔗 HTTP/1.1 (1999)
	RFC 7540 🔗 HTTP/2 (2015)
	RFC 7541 🔗 Header Compression (2, 2015)
	RFC 7230 🔗 Message Syntax and Routing (1.1, 2014)
	RFC 7231 🔗 Semantics and Content (1.1, 2014)
	RFC 7232 🔗 Conditional Requests (1.1, 2014)
	RFC 7233 🔗 Range Requests (1.1, 2014)
	RFC 7234 🔗 Caching (1.1, 2014)
	RFC 7235 🔗 Authentication (1.1, 2014)

HTML Evolution

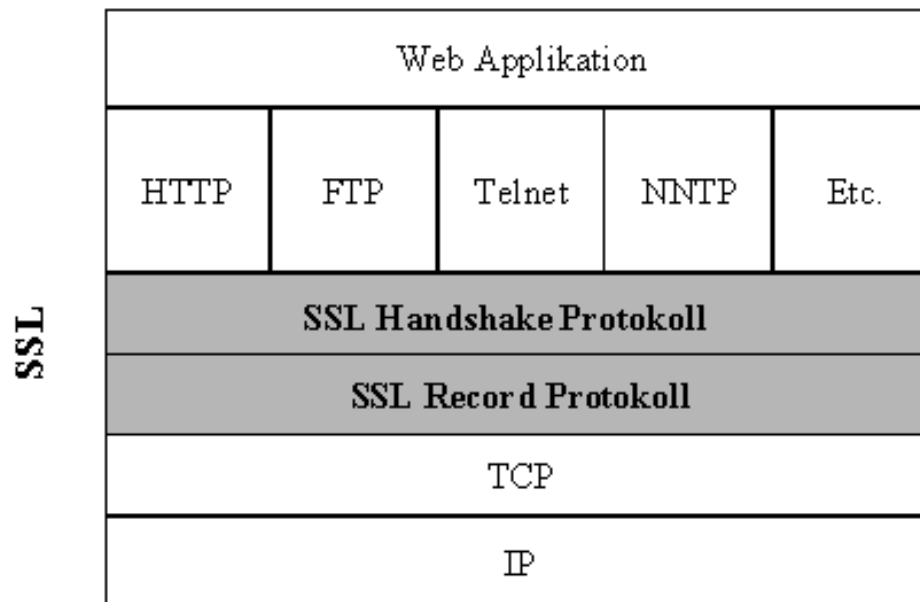
Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	x	x	x	x	x
Images	x	x	x	x	x
Lists	x	x	x	x	x
Active maps & images		x	x	x	x
Forms		x	x	x	x
Equations			x	x	x
Toolbars			x	x	x
Tables			x	x	x
Accessibility features				x	x
Object embedding				x	x
Style sheets				x	x
Scripting				x	x
Video and audio					x
Inline vector graphics					x
XML representation					x
Background threads					x
Browser storage					x
Drawing canvas					x

Absicherung der Kommunikation: https

- Kommunikation kann durch Einsatz des Secure-Socket-Layer-Protokolls (SSL) abgesichert werden
 - SSL wurde von der Firma Netscape entwickelt und ist unter dem Namen Transport Layer Security (TLS) standardisiert (RFC2246)
 - SSL sichert die Transportschicht ab – Basis ist das RSA-Verfahren
 - SSL ist verbindungsorientiert
- Das Secure Socket Layer-Protokoll ist ein hat drei grundlegende Eigenschaften
 1. Eine Verbindung ist privat. Beim Verbindungsaufbau wird ein Sitzungsschlüssel bestimmt, der dann zur Verschlüsselung benutzt werden kann
 2. Die beiderseitige Authentifikation (mutual authentication) wird ermöglicht
 3. Eine Verbindung wird immer durch eine Signatur abgesichert

SSL im ISO-OSI - Stack

- Transparente Schicht zwischen TCP/IP und Application Layer
- Universell einsetzbar

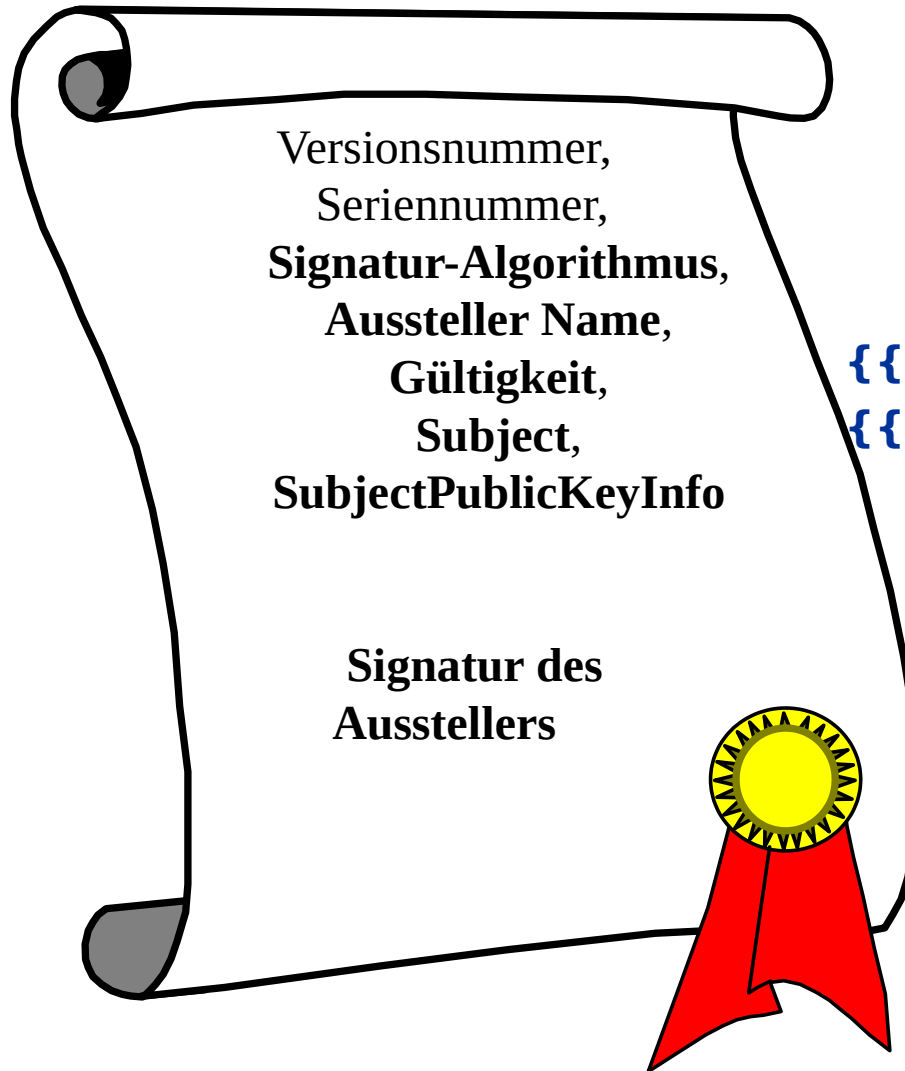


- Zur Verschlüsselung gibt es mehrer Verfahren:
 - symmetrische Verschlüsselung:
Ein Schlüssel zum Ver- und Entschlüsseln
 - asymmetrische Verschlüsselung:
Zwei Schlüssel zum Ver- und Entschlüsseln. Davon ist typischerweise einer öffentlich, und einer privat.
- Das SSL Protokoll muss mehrere Teilaspekte lösen:
- **Authentifizierung**: WER redet mit mir?
- **Tauschen eines gemeinsamen Schlüssels** zur symmetrischen Verschlüsselung

SSL - Handshake Protokoll

- Das SSL Handshake Protokoll erledigt genau diese Aufgaben:
 - Den stärksten gemeinsam unterstützten Algorithmus ermitteln
 - Authentifikation der Kommunikationspartner (Client optional)
 - Ermitteln eines Session Keys zur symmetrischen Verschlüsselung (optional)

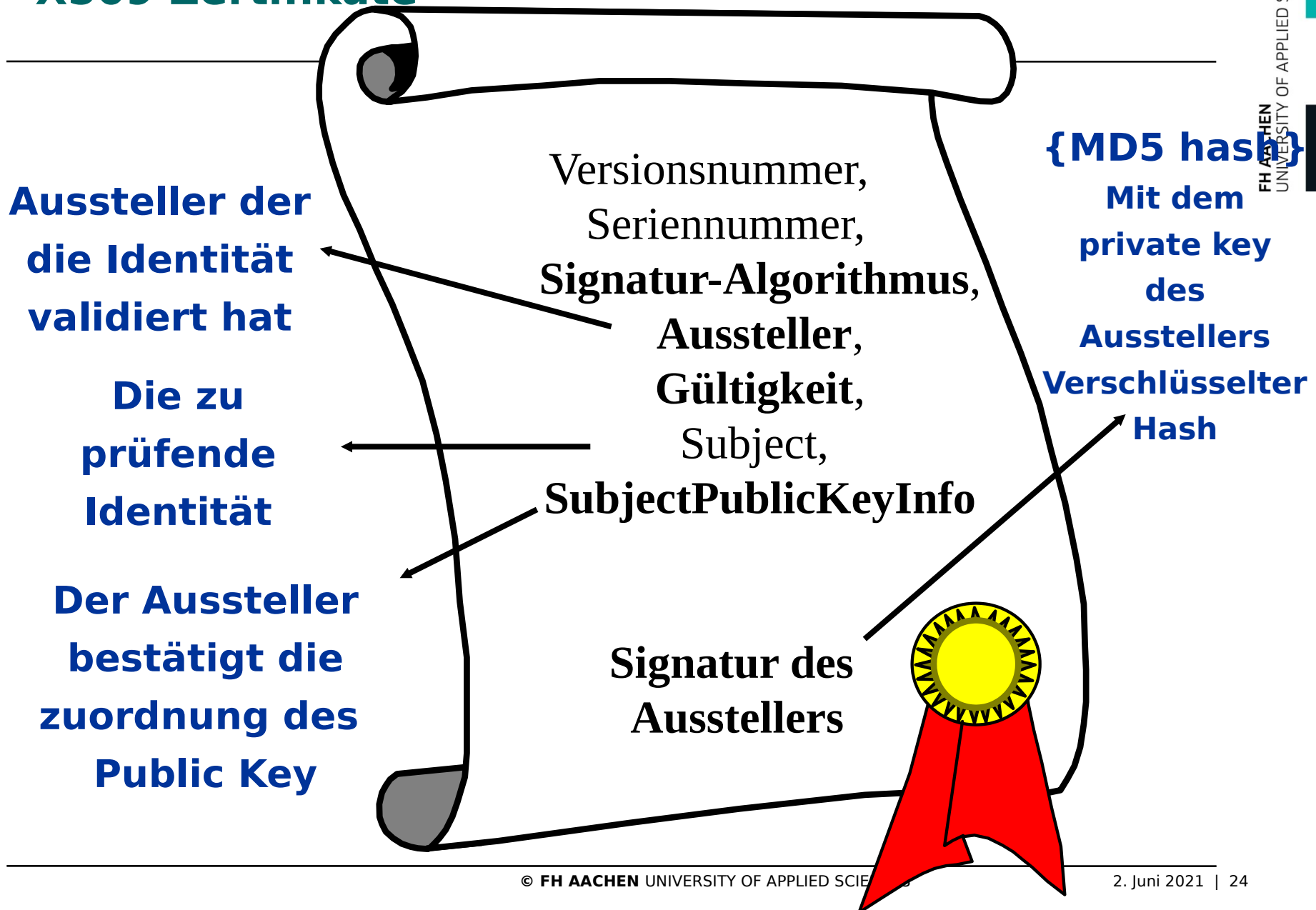
X509 Zertifikate: Wem gehört der public Key?



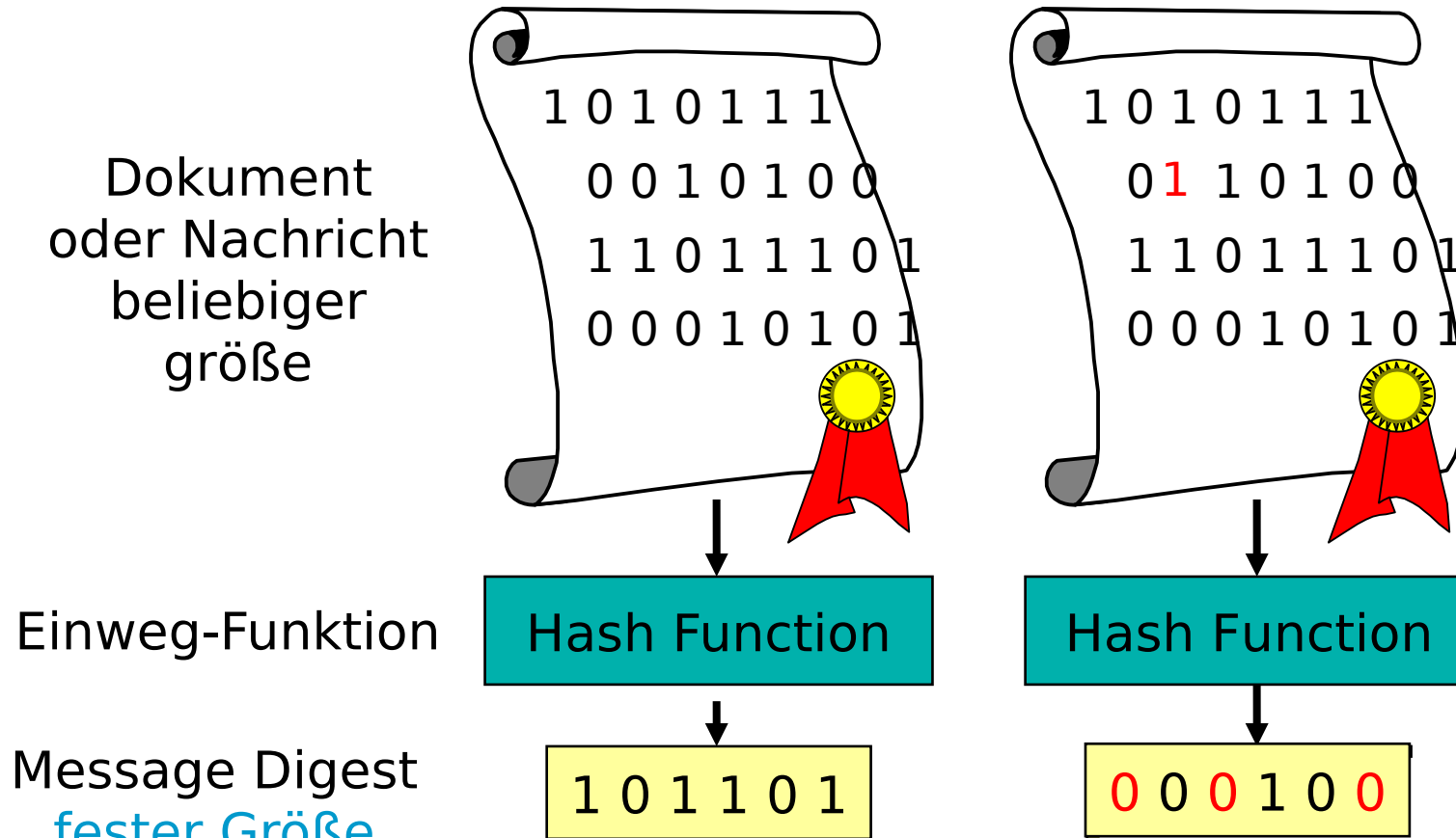
Idee:

$\{\{\text{Klartext}\}\text{private_key}\}\text{public_key} =$
 $\{\{\text{Klartext}\}\text{public_key}\}\text{private_key} =$
Klartext

X509 Zertifikate



Signaturen: Einweg-Hash-Funktionen

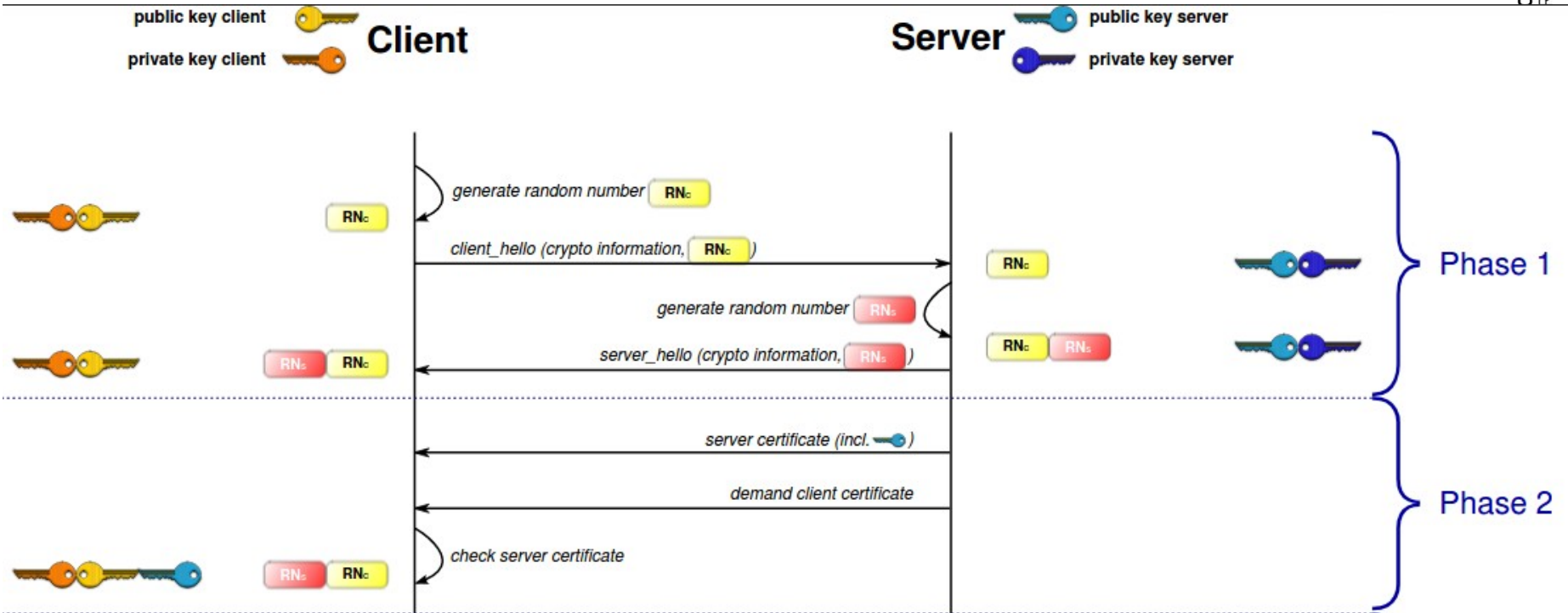


- Die Änderung eines einzelnen Bits im Dokument sollte ungefähr 50% der Hash-bits verändern!

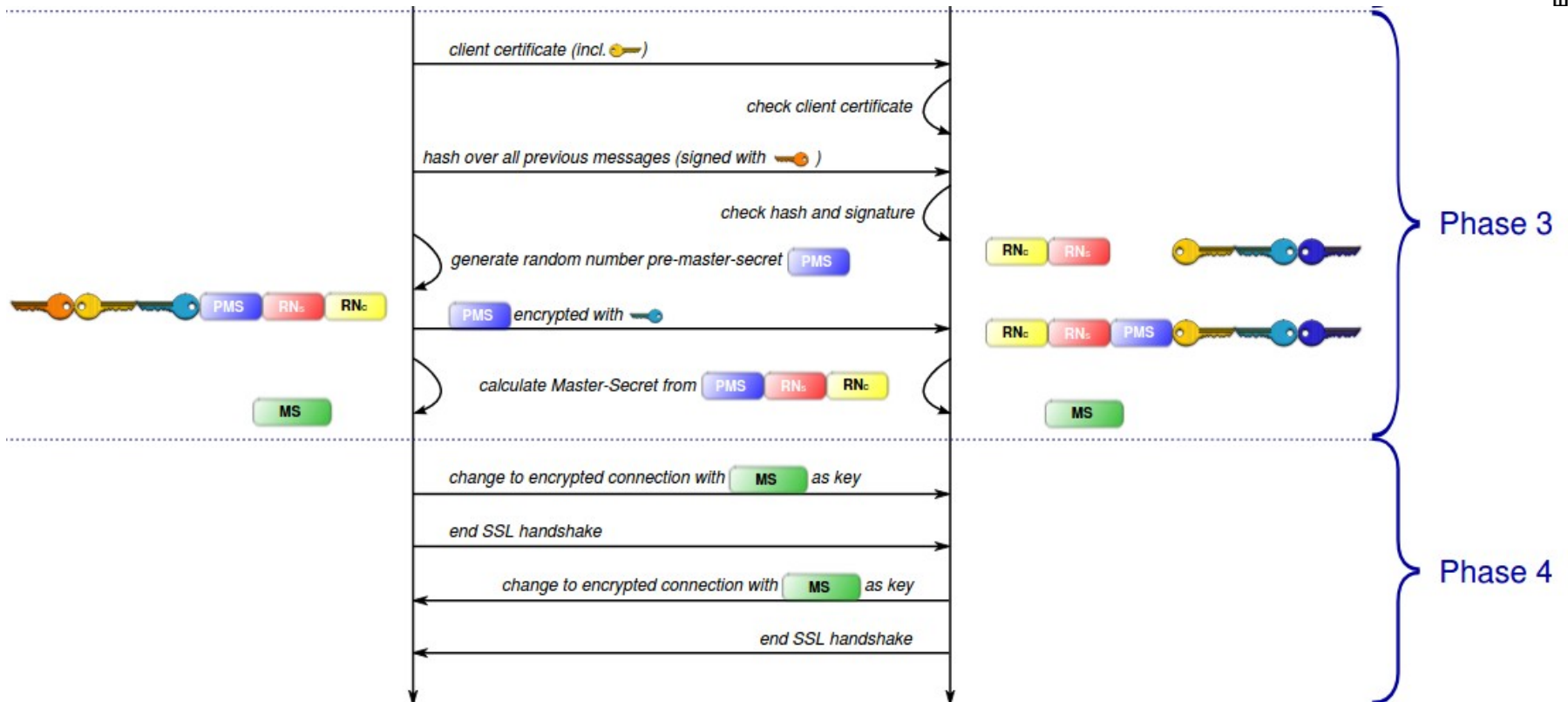
SSL Handshake (vereinfacht)

- Client-Hello-Nachricht:
 - Unterstützte Algorithmen, Zufallsnummer des Clients
 - Session ID (Abkürzung möglich)
- Server-Hello:
 - Ausgewählter Algorithmus, Zufallsnummer des Servers
 - Session ID
- Server Certificates
- Server Hello Done
- Client Key Exchange
 - Nach Prüfung des Zertifikats wird mit dem Public Key des Servers die Basis des Sessions Keys verschlüsselt
- Optional Client Zertifikat + mit Private Key verschlüsselte Zufallsnummer
- Beiderseitiges Finished

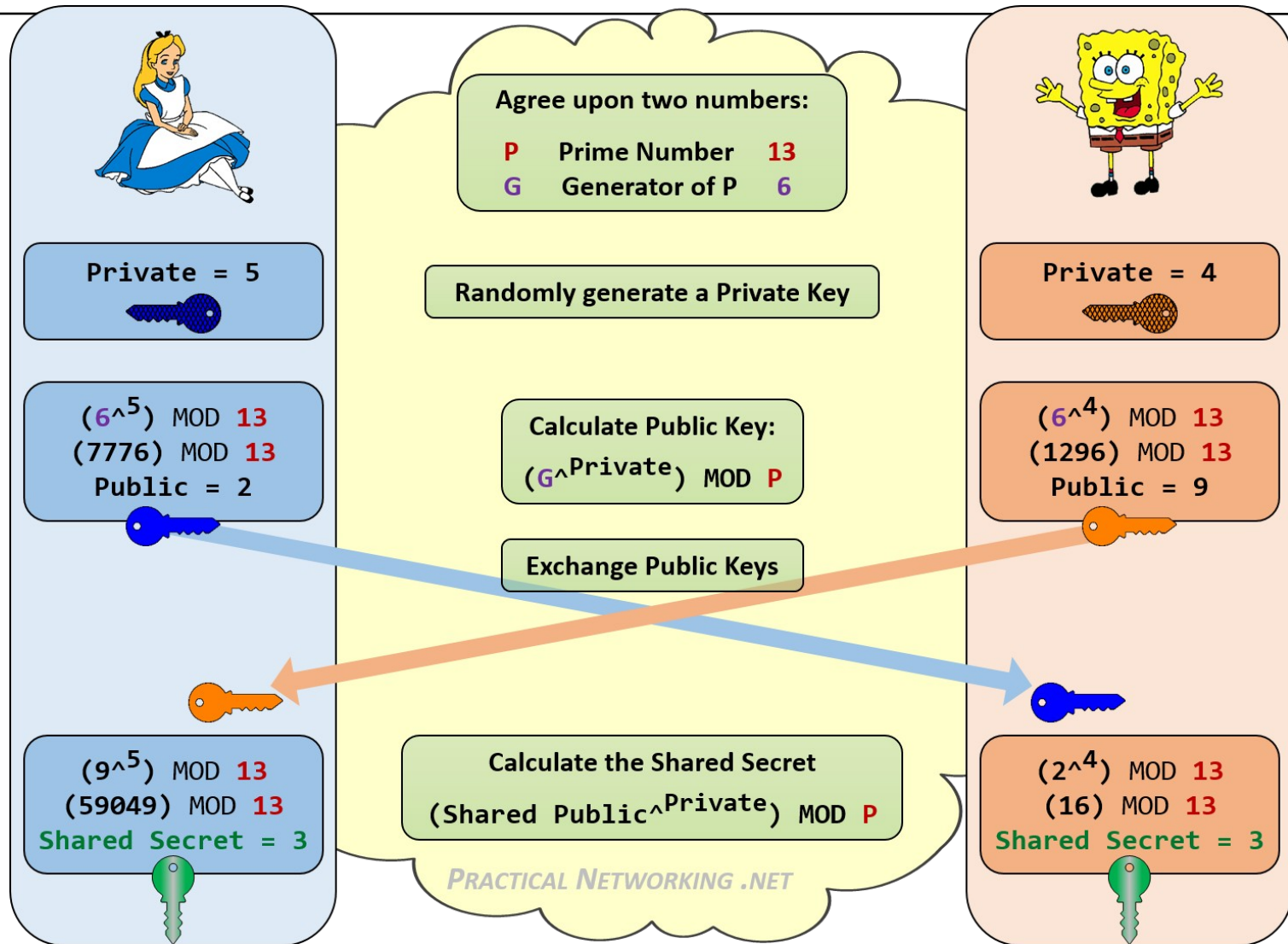
SSL Handshake 1 (vereinfacht)



SSL Handshake 2 (vereinfacht)



Diffie-Hellman-Schlüsselaustausch

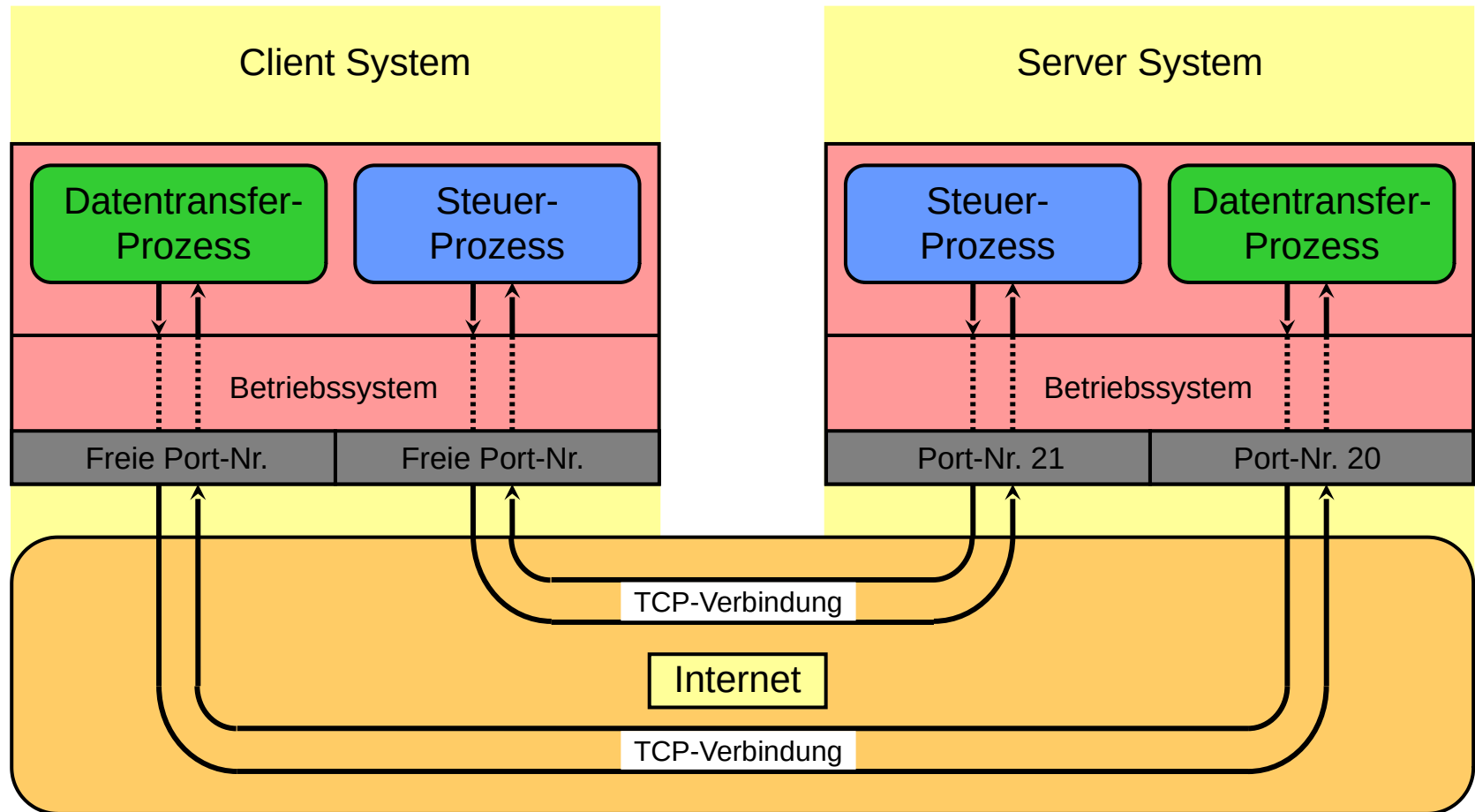


- Vollständig getrennt vom Handshake Protokoll
- Verschickt Daten symmetrisch mit dem im Handshake ausgehandelten Verschlüsselungsalgorithmen und Session Keys
- Bildet zu jedem Datenblock einen Message Digest zur Sicherung der Integrität

FTP - File Transfer Protocol

- FTP ist der *Internet-Standard für die Übertragung von Dateien*
- FTP wird benutzt, um eine **komplette Datei** von einem Rechner auf einen anderen zu **kopieren**
- FTP bietet neben dem reinen File-Transfer noch andere Möglichkeiten:
 - **Interaktiver Zugriff** durch den Nutzer (z.B. Wechsel von Verzeichnissen)
 - **Format-Spezifikation** (Binär- oder Textdateien, ASCII- oder EBCDIC-Code)
 - **Authentifizierung** urspr. nicht empfehlenswert : (login-Name und Passwort)
 - Es gibt Varianten, die eine verbesserte Authentifikation vornehmen

FTP Client/Server-Beziehung



FTP: Passive und Active Mode

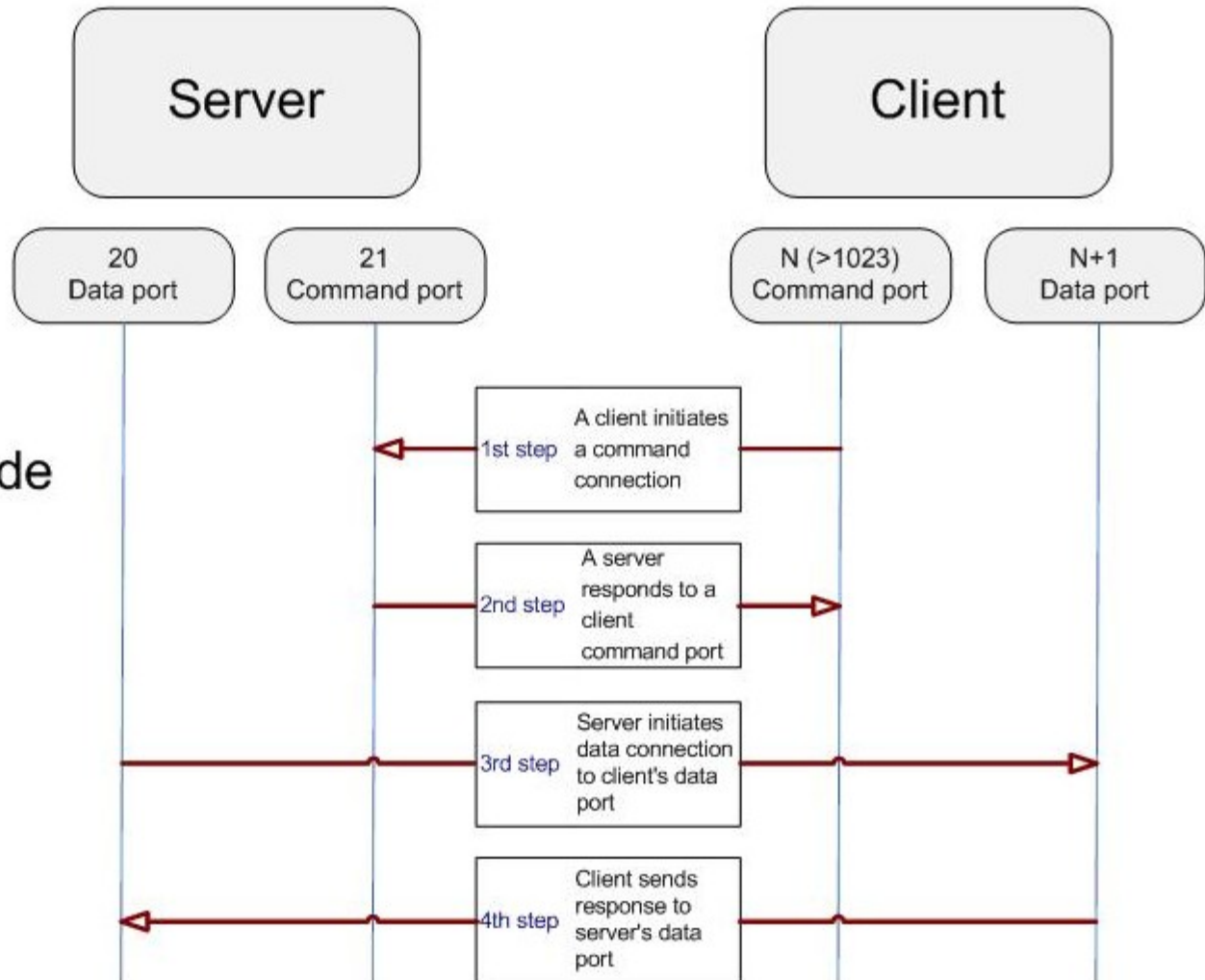
FTP ist deshalb besonders, weil es für die Datenübertragung eine separate TCP-Verbindung verwendet. Diese kann somit über verbesserte Parameter (WSALE-Option verfügen).

Es gibt 2 verschiedene Arten, die Datenverbindung zu öffnen:

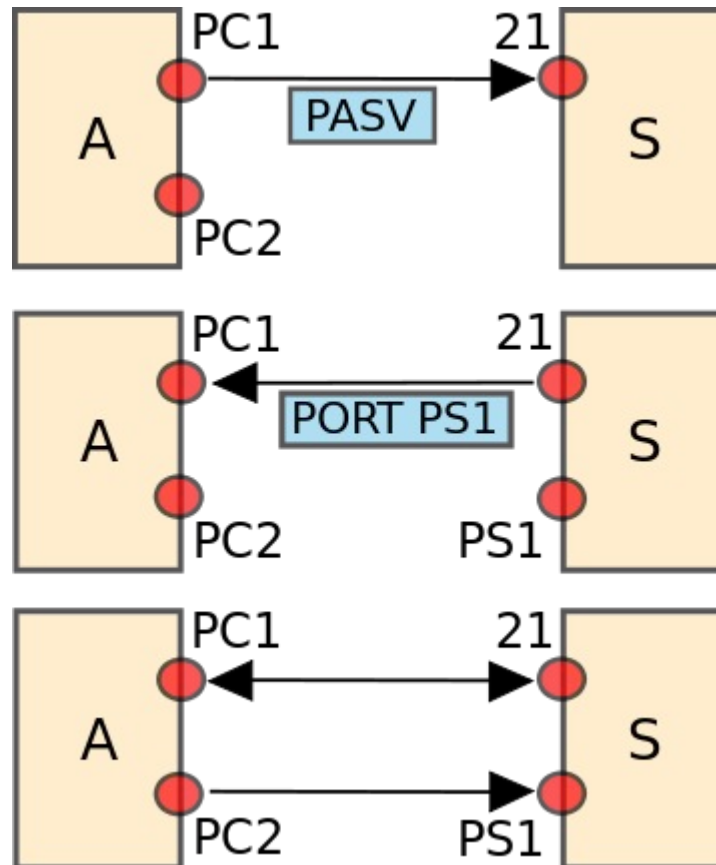
1. **Active Mode:** Hier horcht der Client für die Datenverbindung auf einem zufälligen Port und teilt diesen dem Server über die Kontrollverbindung mit. Hierzu dient das PORT-Kommando. Der Server (Port 20) verbindet sich dann mit dem Client und wickelt den Transfer hierüber ab
2. **Passive Mode:** Öffnet der Server einen Port und teilt diesen dem Client mit. Der Client verbindet sich nun durch mit dem Server. Das PASV-Kommando regelt dies. Vorteil: Dieses Verfahren funktioniert auch bei Adressumsetzungen (NAT) und Firewalls.

FTP: Active Mode

Active FTP Mode



FTP: Passive Mode



https://commons.wikimedia.org/wiki/File:Passive_FTP_Verbindung.svg

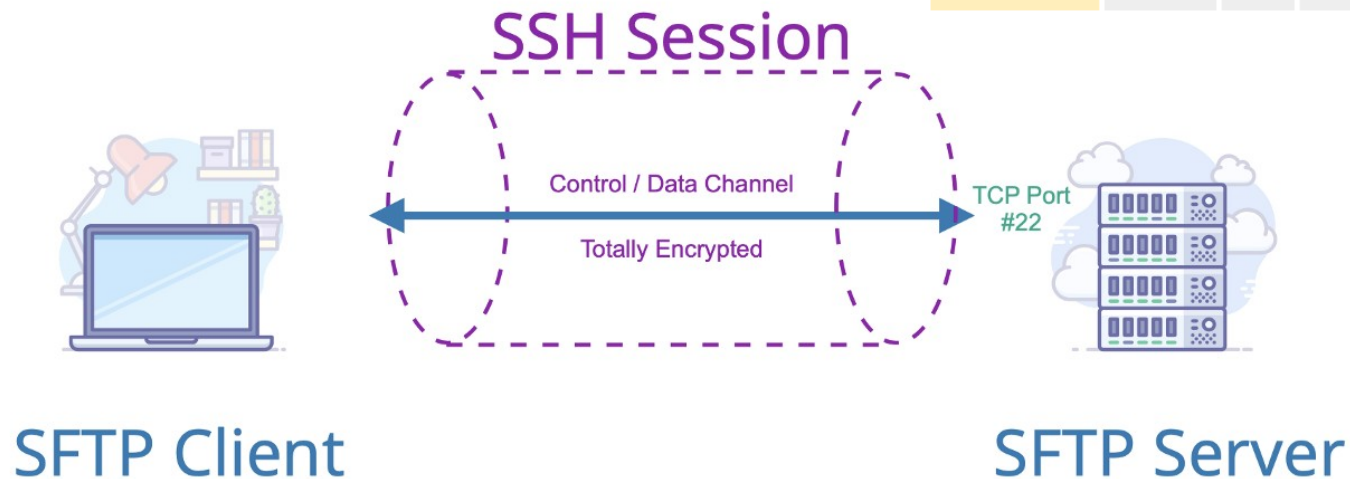
FTP - Befehle

Kommando	Wirkung
open	Verbinden zum FTP-Server
disconnect	Beende die FTP-Sitzung
user	Sende Benutzerinformationen nach dem Verbinden
cd	change directory auf dem entfernten Rechner
lcd	change directory auf dem eigenen Rechner
pwd	Drucke das Arbeitsverzeichnis des entfernten Rechners
get/mget	Der Client empfängt ein (bzw. mehrere) Dokument
put/mput	Der Client sendet ein (bzw. mehrere) Dokument
binary	Setze den Übertragungsmodus auf binary
ascii	Setze den Übertragungsmodus auf ASCII
dir/ls	Liste den Inhalt des entfernten Verzeichnisses auf
help	Hilfe
delete	Lösche eine entfernte Datei
bye	Beende die FTP-Sitzung, Abbruch

Sicheres FTP: SFTP - FTP über SSH

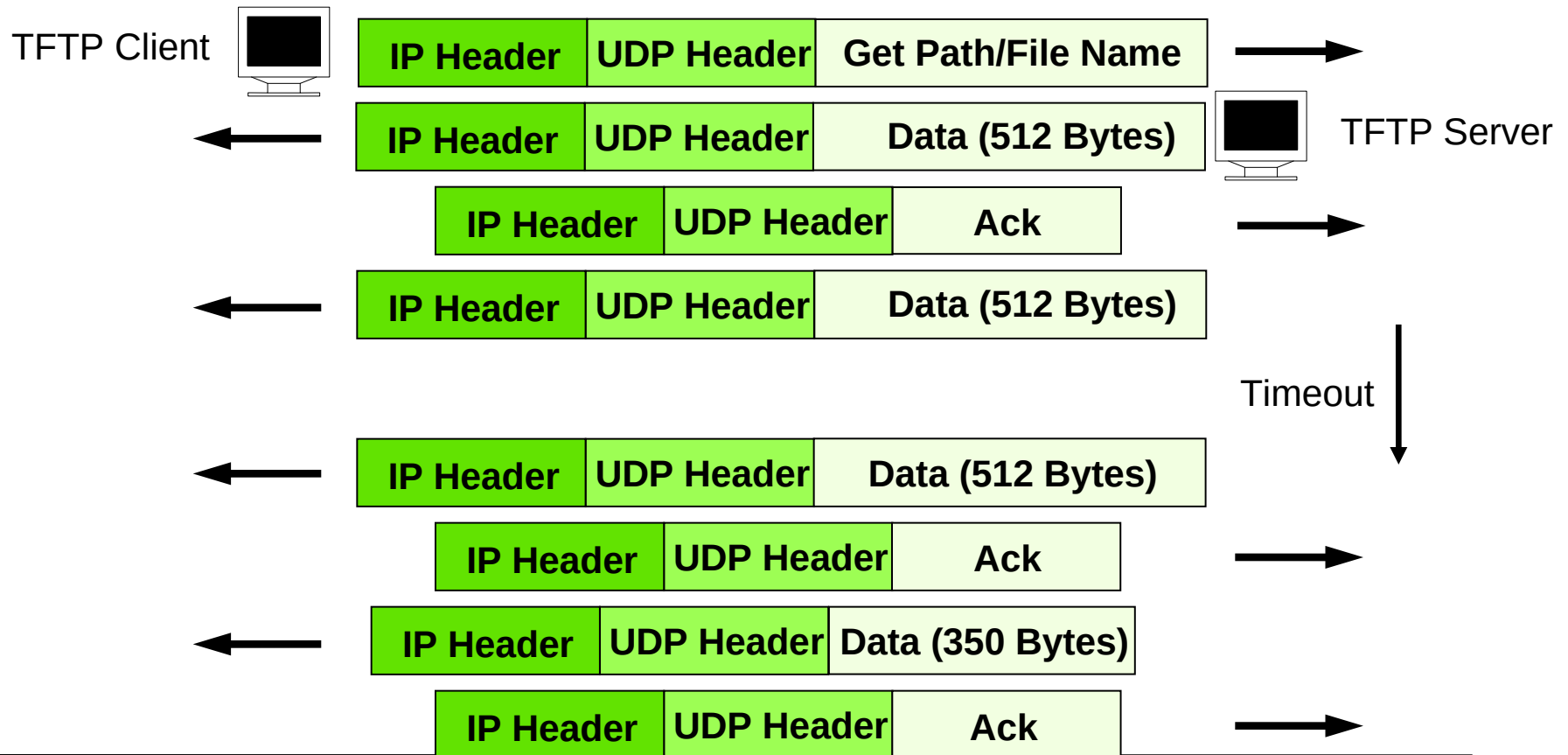
SFTP im TCP/IP-Protokollstapel:

Anwendung	SFTP				
	SSH				
Transport	TCP				
Internet	IP (IPv4, IPv6)				
Netzzugang	Ethernet	Token Bus	Token Ring	FDDI	...



TFTP - Trivial File Transfer Protocol

- TFTP ist ein sehr **einfaches** Protokoll für den File-Transfer
- die Kommunikation läuft über Port 69 und benutzt **UDP**, nicht TCP
- TFTP hat **keine Authentifizierung**
- TFTP benutzt immer 512-Byte-Blöcke

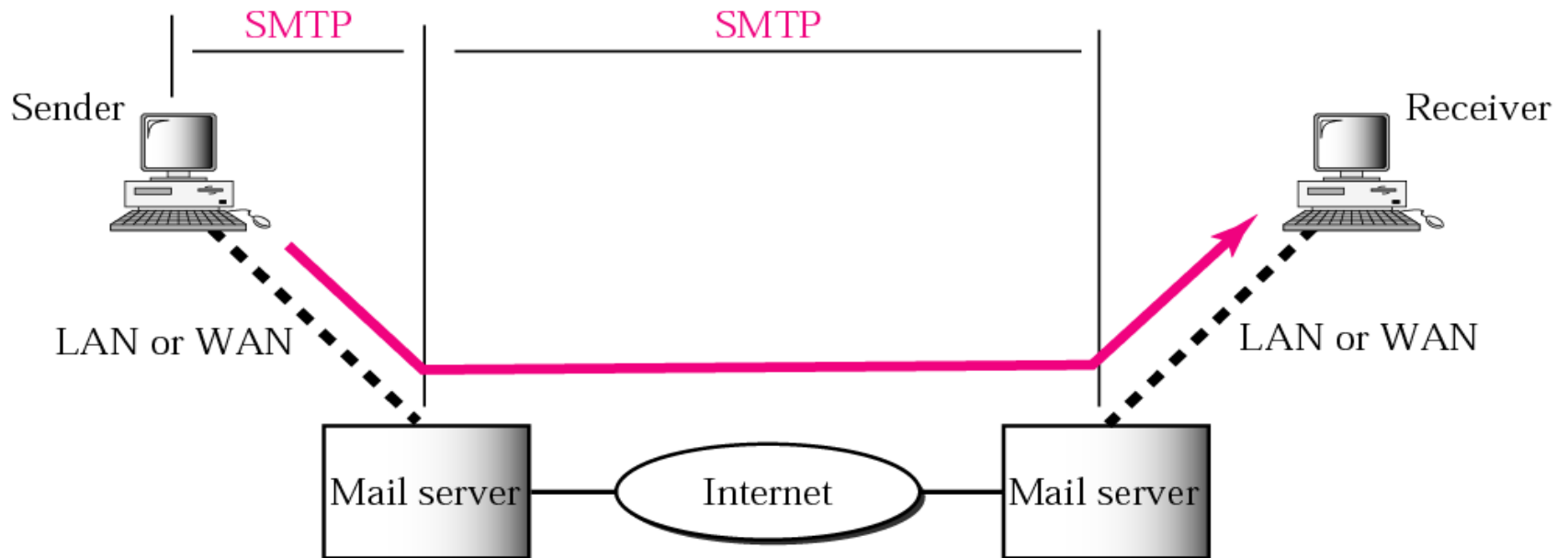


Elektronische Post: E-Mail

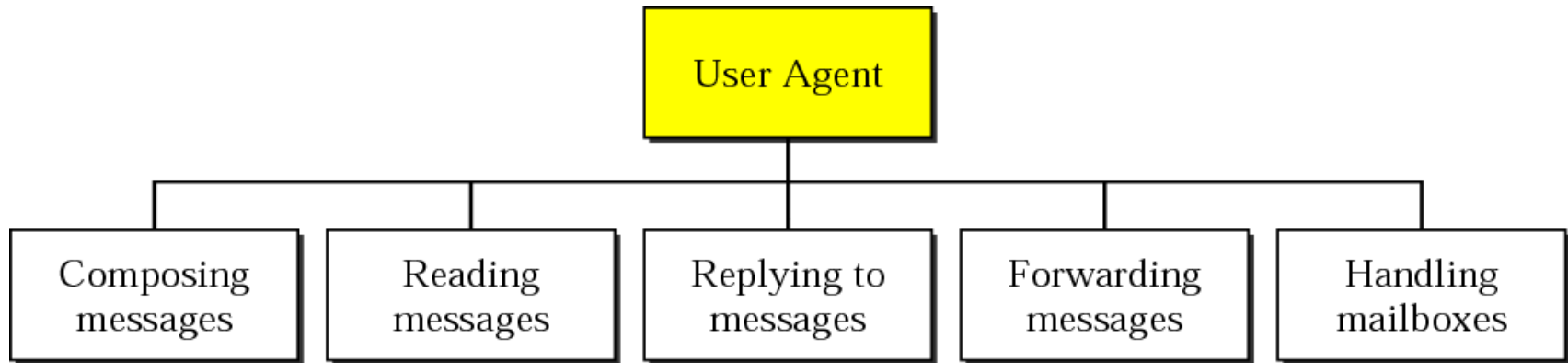
Ein E-Mail-System besteht im allgemeinen aus zwei Subsystemen:

- **User Agent** (UA, normales Email-Programm)
 - läuft meist auf dem Rechner des Benutzers und hilft bei der Bearbeitung von E-Mails
 - Erstellung neuer und Beantwortung alter E-Mail
 - Empfang und Anzeigen von E-Mail
 - Verwaltung von erhaltener E-Mail
- **Message Transfer Agent** (MTA, Mailserver, mailrelay)
 - läuft meist im Hintergrund (rund um die Uhr)
 - Zustellung von E-Mails, die von User Agents losgeschickt wird
 - Zwischenspeicherung von Nachrichten für User oder andere Message Transfer Agents
- **Simple Mail Transfer Protokoll (SMTP)** zum Verschicken von E-Mails

Elektronische Post: E-Mail



Der User Agent



Das Senden von E-Mails

Zum Verschicken einer eMail muss der Benutzer folgende Angaben machen:

- *Nachricht* (meist normaler Text + Attachements, z.B. Word-Datei, GIF...)
- *Zieladresse* (i.a. in der Form mailbox@location, z.B. v.sander@fh-aachen.de)
- evtl. *zusätzliche Parameter* bzgl. Priorität oder Sicherheit

eMail-Formate

Zwei verbreitete Standards:

- **RFC 822** (ARPA Internet Text Messages)
- **MIME** (Multipurpose Internet Mail Extensions)

Bei **RFC 822** besteht die E-Mail aus

- einem einfachen “Umschlag” (erstellt durch den Message Transfer Agent anhand der Daten im Header),
- einer Reihe von Header-Feldern (je eine Zeile ASCII-Text),
- einer Leerzeile und
- der eigentlichen Nachricht (Message Body).

Header einer eMail

Header	Bedeutung
To:	eMail-Adresse des Hauptempfängers (evtl. mehrere oder auch Verteilerliste)
Cc:	Carbon Copy (Durchschrift), Email-Adressen von weniger wichtigen Empfängern
Bcc:	Blind Carbon Copy, Empfänger, die anderen Empfängern <i>nicht</i> angezeigt werden
From:	Person, die die Nachricht generiert hat
Sender:	Adresse des eigentlichen Senders der Nachricht (evtl. ungleich der "From-Person")
Received:	Je ein Eintrag pro Message Transfer Agent auf dem Weg zum Ziel
Return-Path:	Pfad zurück zum Sender (meist nur eMail-Adresse des Senders)
Date:	Sende-Datum und -Uhrzeit
Reply-To:	eMail-Adresse, an die Antworten gerichtet werden sollen
Message-Id:	Eindeutige Nummer der Email (für spätere Referenzen)
In-Reply-To:	Message-Id der Nachricht, auf die geantwortet wurde
References:	Andere relevante Message-Ids
Subject:	Einzeilige Angabe des Inhalts der Nachricht (wird beim Empfänger angezeigt)

Anmerkung: neben dieser Liste existieren noch weitere mögliche Header-Felder

Header einer E-Mail: MIME

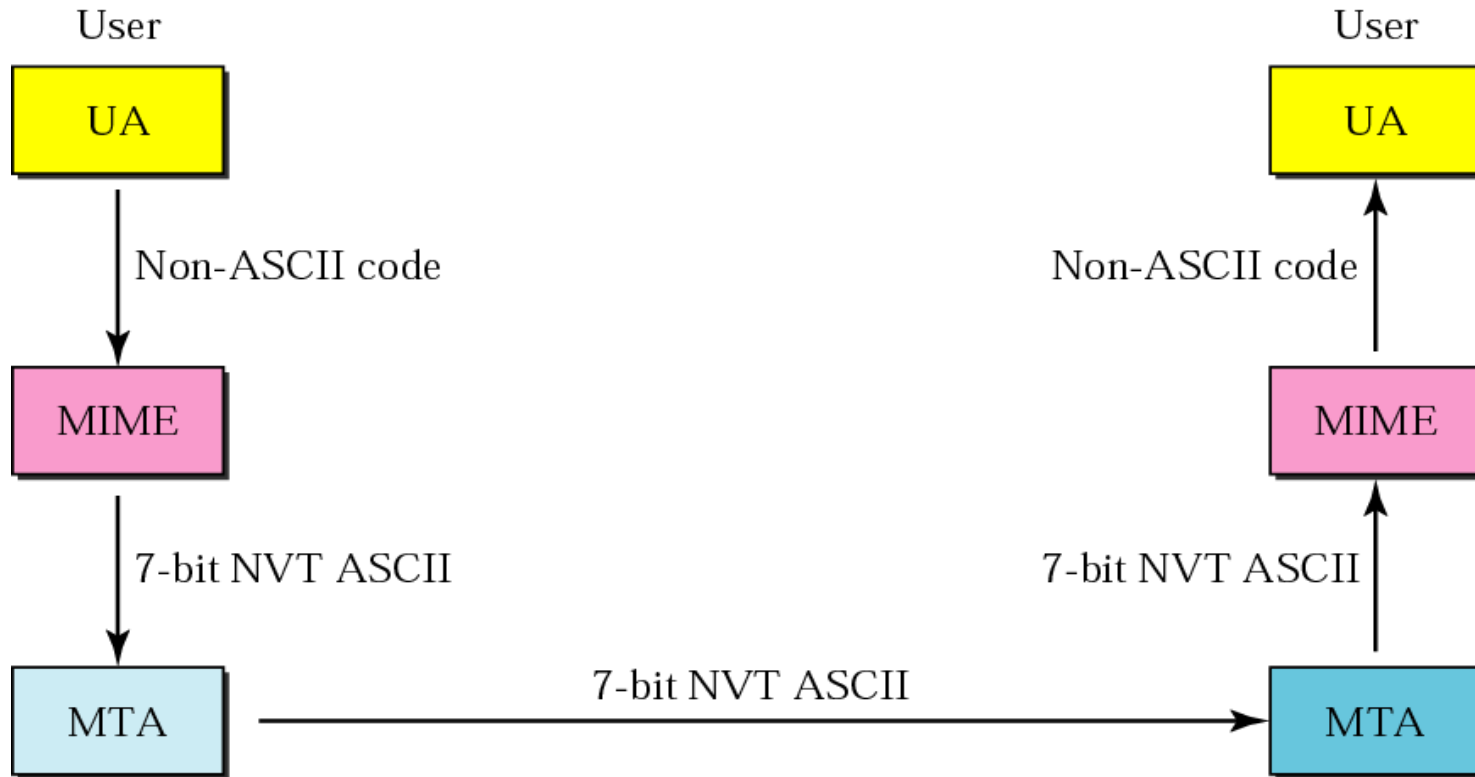
RFC 2822 *nur geeignet für Nachrichten aus reinem ASCII-Text ohne Sonderzeichen*
Heutzutage zusätzlich gefordert:

- eMail in Sprachen mit Sonderzeichen (z.B. französisch oder deutsch)
- eMail in Sprachen, die nicht das lateinische Alphabet benutzen (z.B. russisch)
- eMail in Sprachen, die überhaupt kein Alphabet benutzen (z.B. japanisch)
- eMail, die teilweise überhaupt keinen Text enthält (z.B. Audio oder Video)

MIME behält das RFC 2822 Format bei, definiert dabei aber eine Struktur im Message Body (durch zusätzliche Header) und Kodierungsregeln für Nicht-ASCII-Zeichen

Header	Bedeutung
MIME-Version:	Kennzeichnet die benutzt Version von MIME
Content-Description:	Für Menschen lesbarer String, der den Inhalt der Nachricht beschreibt
Content-Id:	Eindeutiger Bezeichner des Inhalts
Content-Transfer-Encoding:	Verpackung, die für den Inhalt der eMail gewählt wurde (manche Netze verstehen z.B. nur ASCII-Zeichen). Beispiele: base64, quoted-printable
Content-Type:	Typ/Subtyp gemäß RFC 1521, z.B. text/plain, image/jpeg, multipart/mixed

MIME-Ablauf

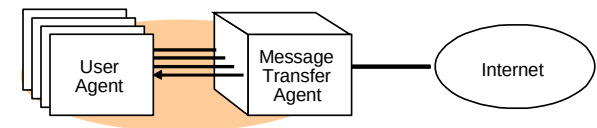
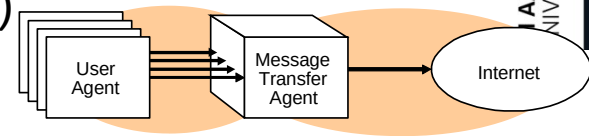


Der konkrete Ablauf wird nunmehr in den RFCs 2045 (MIME) und 2821 (SMTP) beschrieben

eMail Abruf über POP3 oder IMAP

Simple Mail Transfer Protocol (SMTP)

- Versenden von eMails über TCP-Verbindung (Port 25)
- SMTP ist ein einfaches ASCII-Protokoll
- Ohne Prüfsummen, ohne Verschlüsselung
- Ist der Server zum Empfangen bereit, signalisiert er dies dem Client. Dieser sendet die Information, von wem die eMail kommt und wer der Empfänger ist. Ist der Empfänger dem Server bekannt, sendet der Client die Nachricht, der Server bestätigt den Empfang.

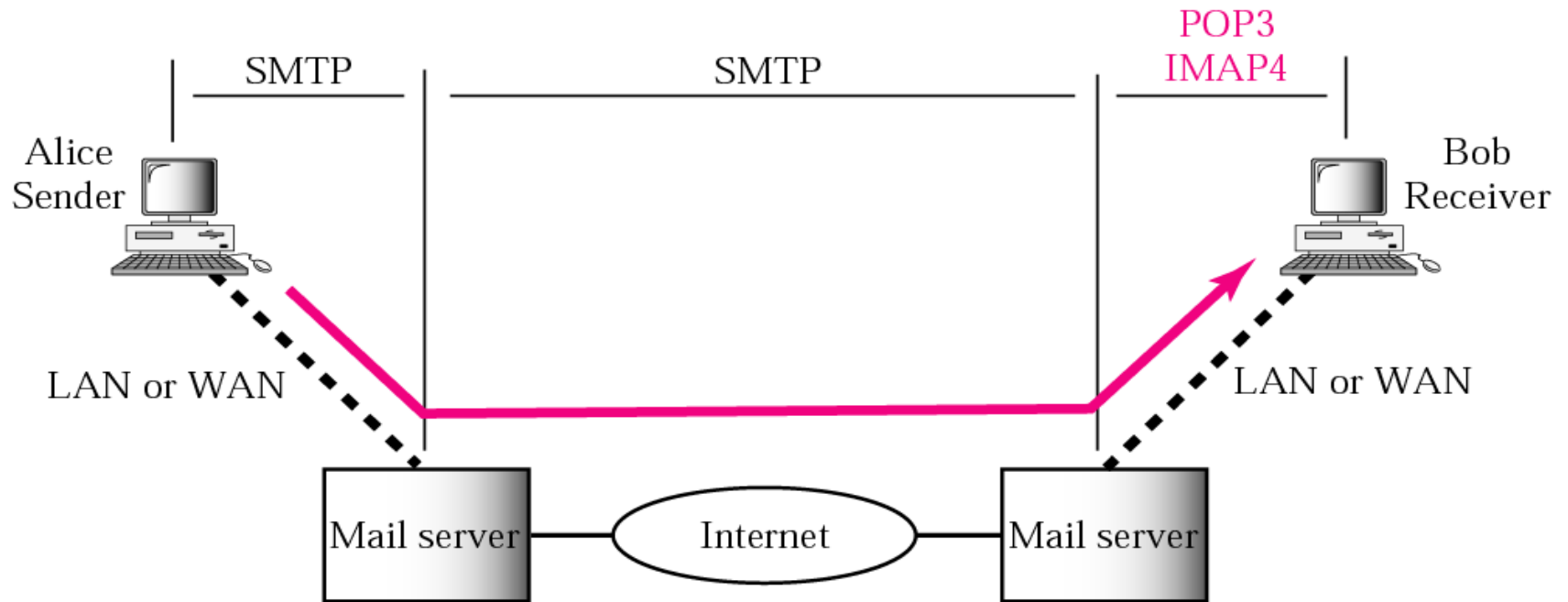


Post Office Protocol Version 3 (POP3)

- Abholen der eMails beim Server über eine TCP-Verbindung, Port 110
- Befehle zum An- und Abmelden, Nachrichten herunterladen, Nachrichten auf dem Server löschen oder liegen lassen, Nachrichten ohne vorherige Übertragung vom Server direkt löschen

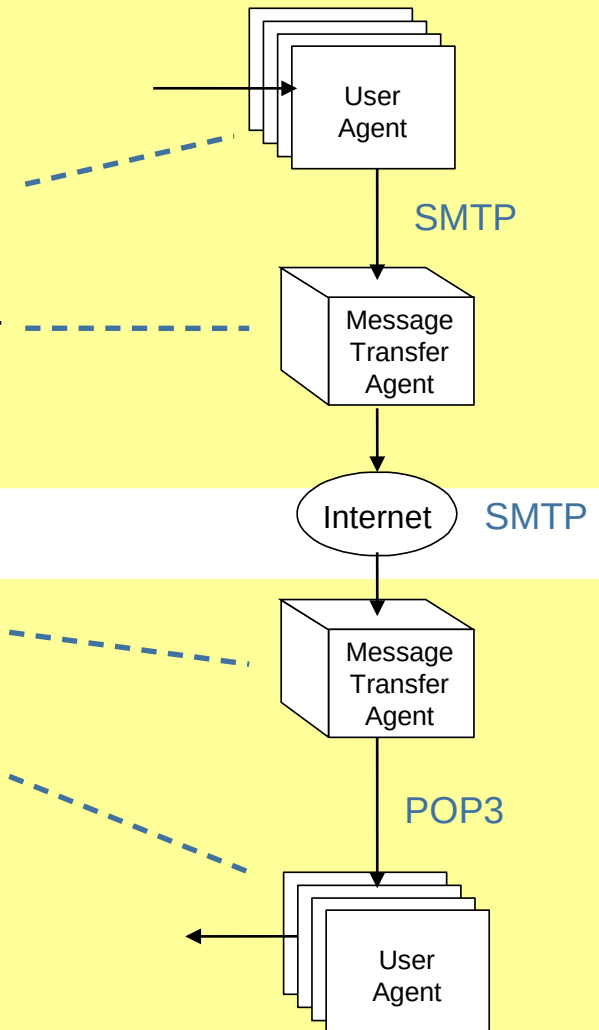
IMAP (Interactive Mail Access Protocol). Hier werden die eMails nicht abgerufen und lokal gespeichert, sondern bleiben auf dem Server liegen!

eMail Abruf über POP3 oder IMAP



Beispiel: eMail über SMTP und POP3

- *Benutzer 1*: schreibt eine eMail
- *Mailprogramm 1 (UA 1)*: Formatiert die eMail, erzeugt die Empfängerliste und schickt die eMail an seinen Mailserver (MTA 1)
- *Client 1 (MTA 1)*: Baut die Verbindung zum SMTP-Server (MTA 2) auf und schickt eine Kopie der eMail dorthin



- *Server (MTA 2)*: Erzeugt den Header der eMail und platziert die eMail in die passende Mailbox
- *Client 2 (UA 2)*: baut die Verbindung zum POP3-Server auf, authentifiziert sich mit Username und Passwort (unverschlüsselt!)
- *Server (MTA 2)*: schickt die eMail an den Client
- *Mailprogramm 2 (UA 2)*: formatiert die eMail
- *Benutzer 2*: liest die eMail

SMTP - Befehlsabfolge

Kommunikation zwischen Partnern (von abc.com nach beta.edu) in Textform der Art:

S: 220 <beta.edu> Service Ready

/ Empfänger ist bereit */*

C: HELO <abc.com>

/ Identifikation des Senders */*

S: 250 <beta.edu> OK

/ Server meldet sich */*

C: MAIL FROM:<Krogull@abc.com>

/ Sender der eMail */*

S: 250 OK

/ Senden ist erlaubt */*

C: RCPT TO:<Bolke@beta.edu>

/ Empfänger der eMail */*

S: 250 OK

/ Empfänger bekannt */*

C: DATA

/ Jetzt kommen die Daten */*

S: 354 Start mail input; end with "<crLf>.<crLf>" on a line by itself

C: From: Krogull @ <crLf>.<crLf>

/ ab hier normales
Nachrichtenformat */*

S: 250 OK

C: QUIT

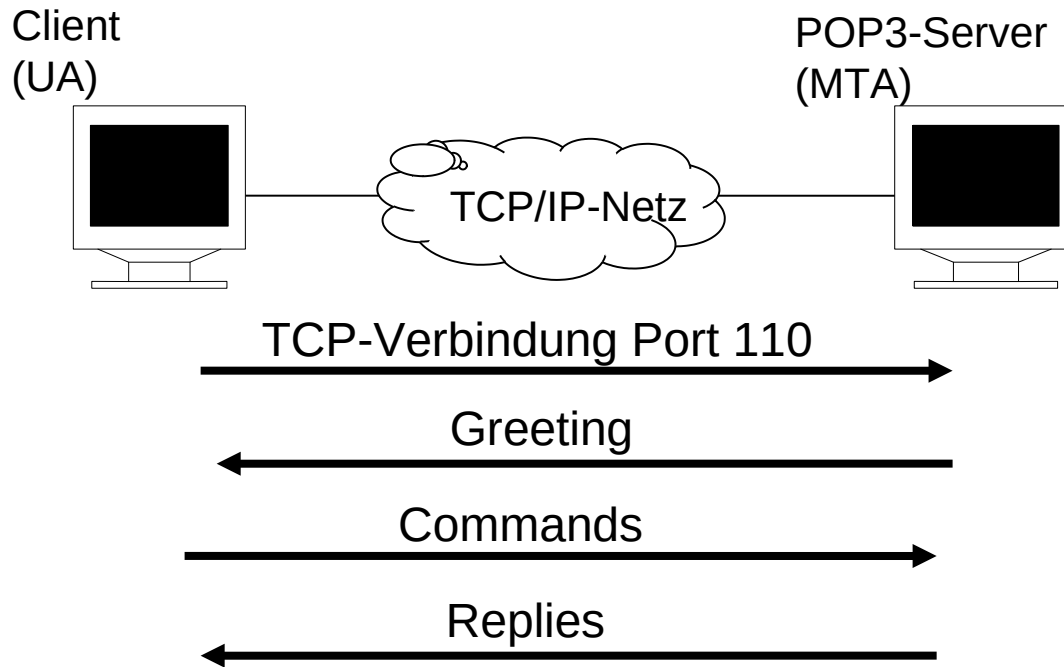
/ Beenden der Verbindung */*

S: 221 <beta.edu> Server Closing

S = Server, empfangender MTA / C = Client, sender MTA

POP3-Prozess

Abholen der eMails vom Server mittels POP3:



- Authorisierungsstatus:
USER name

PASS string
- Transaktionsstatus
STAT
LIST [msg]
RETR msg
DELE msg
NOOP
RSET
QUIT


- Damit werden die vollständigen eMails vom Server auf den Client transferiert
- Wahlweise können hierbei die Nachrichten auch gelöscht werden

Idee: Zugriff auf Server-Verzeichnisse wie auf lokale Verzeichnisse

- Daten verbleiben auf dem Server. Alle Aktionen führt der Client durch
- Das Protokoll ist komplexer als bei POP3 und erlaubt die Manipulation von Mailverzeichnissen auf dem Server (Erstellen, Umbenennen, Löschen; Setzen/Löschen von Flags; Suchen von Mails)
- Reduziert die zu übertragenen Daten, da zunächst nur die Nachrichten-Header übertragen werden
- Offline-Betrieb und Resynchronisation mit dem Server möglich
- Spezifiziert in **RFC 3501**

Mit IMAP verbleiben die eMail auf dem Server

Wie kann ich meine E-Mails über eine gesicherte Verbindung (SSL oder TLS) versenden und empfangen?

 [Druckansicht](#)
[FAQ #118](#)

Möchten Sie E-Mails mit Ihrem eigenen E-Mail Programm verschlüsselt versenden und empfangen, aktivieren Sie bitte die entsprechende Option in den Konto-Einstellungen. Diese Einstellungsmöglichkeit finden Sie in der Regel in Ihrem E-Mail Programm unter dem Menüpunkt **Extras > E-Mail-Einstellungen** bzw. **Konten** oder **Kontoeinstellungen**. Die Option selbst wird unterschiedlich benannt. Bei Outlook / Outlook Express z. B. heißt der Menüpunkt **Server erfordert eine verschlüsselte Verbindung (SSL)**, in Mozilla Thunderbird ist die **Verbindungssicherheit: SSL/TLS** einzustellen. Bitte lesen Sie dies, falls notwendig, in der Dokumentation zu Ihrer Software nach.

Die notwendigen E-Mail-Server lauten wie folgt:

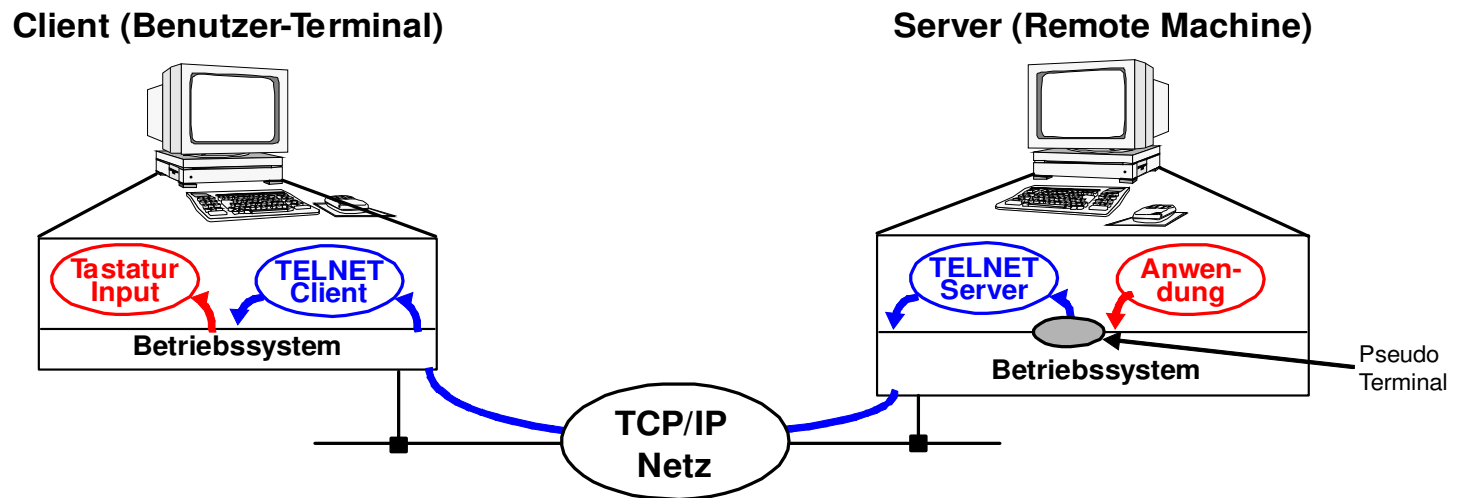
Posteingang (POP3):	pop3.strato.de
Posteingang (IMAP):	imap.strato.de
Postausgang (SMTP):	smtp.strato.de

Sofern Ihr E-Mail Programm Portangaben benötigt, lauten diese:

Protokoll	Port
POP3 (SSL/TLS)	995
IMAP (SSL/TLS)	993
SMTP (SSL/TLS)	465

Telnet - Entferntes Arbeiten

- TCP ermöglicht den transparenten, interaktiven Gebrauch von „entfernten“ Maschinen
- verbreitetes Protokoll: TELNET, welches auf einer Client/Server-Kommunikation basiert
- Ein „Pseudo-Terminal“ des Servers interpretiert Zeichen, als kämen sie von der eigenen Tastatur



- bei Antwort des Servers umgekehrter Weg (Pseudo-Terminal fängt Antwort ab, leitet sie über TCP an den Client weiter, der die Ausgabe am Bildschirm macht)
- **Benutzername und Passwort werden unverschlüsselt übertragen**

rlogin und rsh als Alternative zu Telnet (für Unix)

- *rlogin* ist eine sehr flexible Alternative zu Telnet. Sogenannte Trusted Hosts können sich login-Name und Zugriffsrechte auf Dateien teilen.
- Vorteile gegenüber TELNET:
 - Bei rlogin auf einem Trusted Host entfällt die Abfrage des Passworts.
 - Da ausschließlich unter Unix verwendet, vereinfacht sich die Kommunikation zwischen Client und Server: beide Seiten kennen so etwas wie Standard Input und Output, Standard Error ...
 - Umgebungsvariablen des Benutzers (z.B. Terminaltyp) werden automatisch übertragen, so dass entfernte Sitzungen große Ähnlichkeit mit lokalen Sitzungen haben.
- *rsh* ist eine Variante von rlogin:
 - Ziel: Auf einfache Art und Weise einzelne Kommandos auf der Remote Machine ausführen (`rsh machine command`).
 - Automatische Authentifizierung erlaubt die Benutzung nicht nur interaktiv, sondern auch aus Programmen heraus (ohne Passwortabfrage).

- **ssh** adressiert die Sicherheitsprobleme von telnet und rlogin. Es ist ein Protokoll zur Erstellung einer sicheren Verbindung zwischen zwei Systemen. Alle während der Verbindung gesendeten und empfangenen Daten werden mit einer 128 Bit-Verschlüsselung verschlüsselt.
- ssh unterstützt verschiedene Authentisierungsarten:
 - Bei der so genannten hostbased-Authentifizierung akzeptiert ein Rechner ohne eigene account-spezifische Tests die Vorgaben eines fremden Rechners. Es wird höchstens die Identität des fremden Rechners überprüft.
 - Die Authentifikation mit einem Passwort ist derzeit die "übliche" Methode, um sich an einem Rechner anzumelden. Die Sicherheit dieses Mechanismus beruht auf der Geheimhaltung des Passwortes, dessen Übertragung allerdings verschlüsselt wird
 - Um auch das Übertragen eines verschlüsselten Passwortes zu vermeiden, werden die so genannten public-key-Verfahren eingesetzt

SSH: Port-Forwarding

- Port-Forwarding:
 - verschlüsselte Verbindung zwischen zwei beliebigen Ports
 - kann auch ohne Shell genutzt werden
 - lokaler Port führt direkt auf den Zielport, als wäre dieser lokal
- universell einsetzbar, u. a.:
 - FTP sicherer machen (Tunneln des Kommando-Ports)
 - POP3/SMTP (Mailversand) sicherer machen
 - X Window Datenverkehr absichern
- Aufruf:
 - **ssh -L port:zielHost:zielPort <Rechner>** leitet *localhost:port* via *Rechner* zu *zielHost:zielPort* weiter
 - **ssh -R port:zielHost:zielPort <Rechner>** leitet *Rechner:port* via *localhost* zu *zielHost:zielPort* weiter

SNMP (Simple Network Management Protocol)

Die Objekte des Internet-Managements sind Rechner und vor allem Router

Ähnlich wie bei SMTP wird das Internet-Management durch zwei unabhängige standardisierte Teilbereiche beschrieben:

- Das Protokoll SNMP, das festlegt, wie Management-Information kommuniziert wird (Formate und Bedeutung von SNMP-Nachrichten) und
- die Spezifikation der Daten (MIB \times *Management Information Base*).
Die MIB spezifiziert die Informationseinheiten (*items*), die vorgehalten werden müssen, und welche Operationen darauf erlaubt sind.

Wie bei den anderen Anwendungsdiensten, funktioniert auch das Management nach dem *Client/Server*-Prinzip.

In jedem Objekt (vor allem Router) muss ein *Server* installiert sein, der die in der MIB spezifizierten Informationen sammelt, diese gegebenenfalls einem *Client* zur Verfügung stellt (per SNMP) und von einem *Client* Kommandos entgegennimmt.

Für das Ausführen von Management-Funktionen ist eine Authentifizierung erforderlich ist.

SNMP (Simple Network Management Protocol)

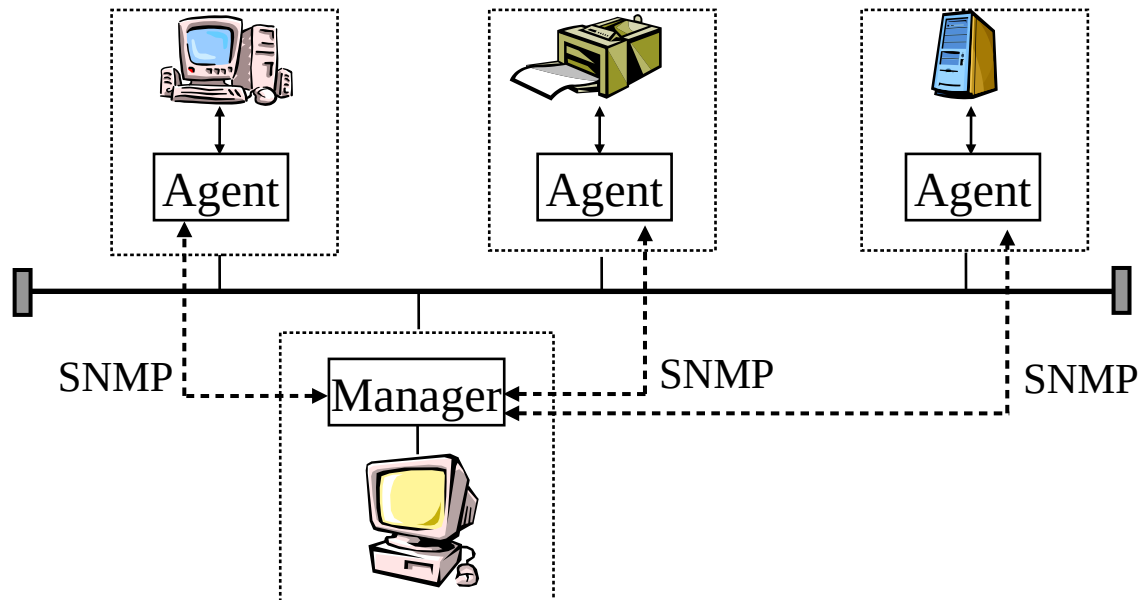
Verwaltete Ressourcen integrieren **SNMP-Agenten** (Software-Prozess)

Die Agenten verwalten die Managementinformationen der Komponente

z.B. Anzahl eingegangener/verlorener Pakete

Der **Manager** (Software-Prozess) dient der Kommunikation mit den Agenten

Protokoll: SNMP (verwendet UDP)



FH Aachen
Fachbereich 9 Medizintechnik und Technomathematik
Prof. Dr.-Ing. Andreas Terstegge
Straße Nr.
PLZ Ort
T +49. 241. 6009 53813
F +49. 241. 6009 53119
Terstegge@fh-aachen.de
www.fh-aachen.de