Rechnernetze:
(3) Anwendungsschicht

Dr. Christian Keil

Gliederung der Vorlesung

- Einführung und Historie des Internets
- Schichtenmodell
- Netzwerk als Infrastruktur
- Layer 7: Anwendungsschicht
  - Verschiedene Anwendungsprotokolle
- Layer 7/4: Socketprogrammierung
- Layer 4: Transportschicht
- Layer 3: Netzwerkschicht
- Layer 2: Sicherungsschicht

Inhalte dieses Kapitels

In diesem Kapitel betrachten wir beispielhaft einige wenige Protokolle der Anwendungsschicht (Layer 7).

Wir konzentrieren uns dabei auf Unterschiede in der Gestaltung dieser Protokolle, wie sie z. B. auch bei der Analyse mit Netzwerk-Sniffern schnell sichtbar werden.

Ziele dieses Kapitels

Sie können grundlegende Protokolle für Internet-Anwendungen benennen und gegeneinander abgrenzen.

Sie haben verstanden, dass zum konkreten Verständnis eines Protokolls das Lesen eines RFCs notwendig ist, wobei Ihnen spätestens bei der Übungsaufgabe 2 klar werden soll, dass auch bei auf den ersten Blick einfachen Protokollen interessante Herausforderungen versteckt sein können.

Die Welt der E-Mail!

… und ist E-Mail nun:

Synchron oder Asynchron?

Verbindungslos oder Verbindungsorientiert?

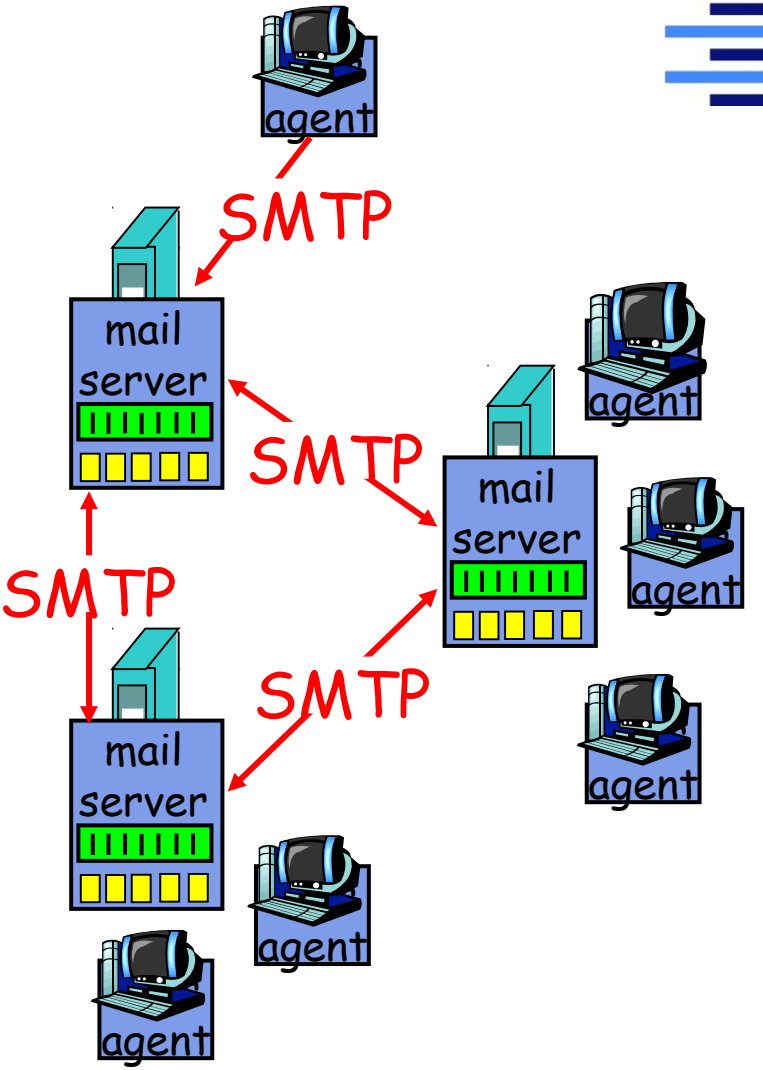
Unicast oder Broadcast?

Client-Server oder Peer-to-Peer?

Electronic Mail

Hauptkomponenten:

- User agents
- Mailserver
- SMTP: Simple Mail Transfer Protocol

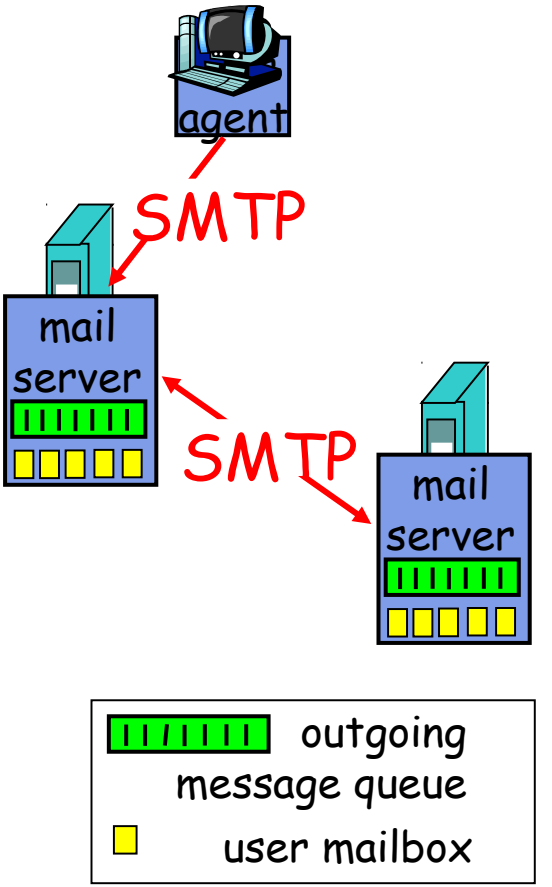


Electronic Mail: User agents

- **“Mail Reader”-Programme, z. B. Outlook, Thunderbird, haben folgende Funktionen:**
  - Verfassen, Bearbeiten, Lesen, Speichern von Mailnachrichten
  - Senden von Nachrichten zu einem Mailserver (per SMTP)
  - Holen von Mails vom Mailserver, die dort ab Eingang gespeichert werden (per POP3 oder IMAP)

Electronic Mail: Mailserver

- **Mailbox:**
  - enthält eingegangene Nachrichten für Benutzer
- **Message Queue:**
  - Warteschlange von zu sendenden Nachrichten



Electronic Mail: Mailserver

- **SMTP-Protokoll:**
  - Zwischen User Agent (Client) und Mailserver (Server) zum Senden von Nachrichten
  - Zwischen Mailservern (der sendende als Client, der empfangende als Server) zum Weiterleiten von Nachrichten untereinander (store and forward)

SMTP

RFC 821 definiert das Simple Mail Transfer Protocol (SMTP, 25/tcp)

- Kleiner Befehlssatz
- Austausch von (ASCII-) Textnachrichten im Store-and-forward Prinzip
- Binärdaten (Bilder, Programmdateien, Audio, etc.) müssen in ASCII konvertiert werden.
- Signalisierungsstandard MIME: Multipurpose Internet Mail Extensions

SMTP (2)

- Eine E-Mail besteht aus einem Envelope mit Vermittlungsdaten (env-to) und Einträge der Vermittlungsserver sowie der eigentlichen Nachricht im Body
- Die im Mail-Browser lesbaren Angaben zu Sender (from), Empfängern (to/cc), Subject sind Teil der transportierten Nachricht, also des Bodies!

SMTP ist unsicher

- **SMTP bietet keine Gewährleistung von Geheimhaltung oder Integrität**
  - Spoofing ist immer möglich
- **ESMTP (Extended SMTP)**
  - Aushandlung einer Authentifikation
    - SMTP-After-POP (vor dem Senden Post abholen)
    - SMTP-Auth (Benutzername + Passwort)
    - SMTPS / StartTLS (Verschlüsselung mit SSL/TLS)

SMTP Beispiel-Sitzung

```
S: 220 smtp.haw-hamburg.de
C: HELO laptop.dsl.hamburg.de
S: 250 Hello laptop..., pleased to meet you
```

SMTP Beispiel-Sitzung

```
S: 220 smtp.haw-hamburg.de
C: HELO laptop.dsl.hamburg.de
S: 250 Hello laptop..., pleased to meet you
C: MAIL FROM: <alice@laptop.dsl.hamburg.de>
S: 250 alice@laptop.dsl.hamburg.de... Sender ok
```

SMTP Beispiel-Sitzung

```
S: 220 smtp.haw-hamburg.de
C: HELO laptop.dsl.hamburg.de
S: 250 Hello laptop..., pleased to meet you
C: MAIL FROM: <alice@laptop.dsl.hamburg.de>
S: 250 alice@laptop.dsl.hamburg.de... Sender ok
C: RCPT TO: <kpk@kossakowski.de>
S: 250 kpk@kossakowski.de ... Recipient ok
```

SMTP Beispiel-Sitzung

```
S: 220 smtp.haw-hamburg.de
C: HELO laptop.dsl.hamburg.de
S: 250 Hello laptop..., pleased to meet you
C: MAIL FROM: <alice@laptop.dsl.hamburg.de>
S: 250 alice@laptop.dsl.hamburg.de... Sender ok
C: RCPT TO: <kpk@kossakowski.de>
S: 250 kpk@kossakowski.de ... Recipient ok
C: DATA
S: 354 Enter mail,end with "." on a line by itself
C: [... header and body of Email here ...]
```

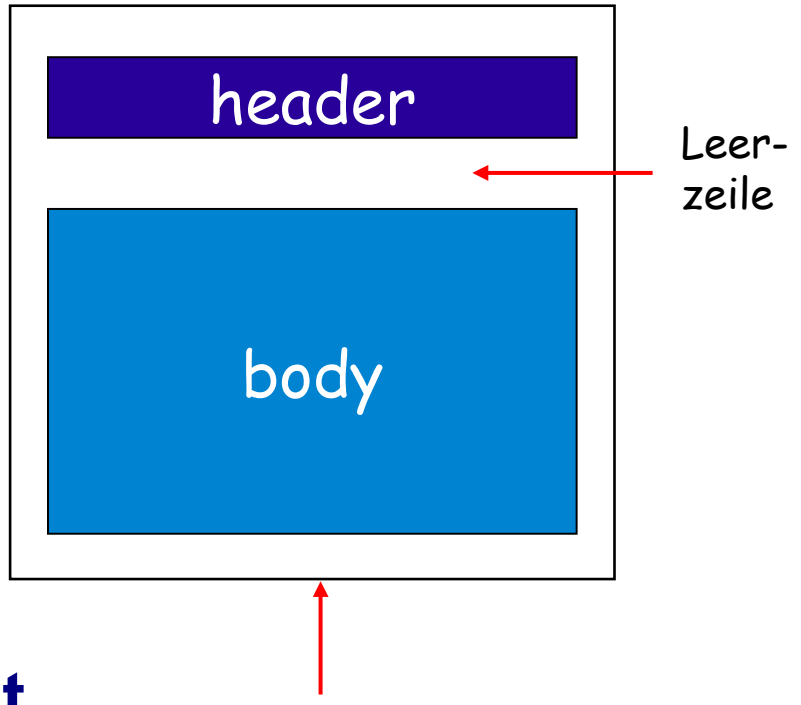
SMTP Beispiel-Sitzung

```
S: 220 smtp.haw-hamburg.de
C: HELO laptop.dsl.hamburg.de
S: 250 Hello laptop..., pleased to meet you
C: MAIL FROM: <alice@laptop.dsl.hamburg.de>
S: 250 alice@laptop.dsl.hamburg.de... Sender ok
C: RCPT TO: <kpk@kossakowski.de>
S: 250 kpk@kossakowski.de ... Recipient ok
C: DATA
S: 354 Enter mail,end with "." on a line by itself
C: [... header and body of Email here ...]
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 smtp.haw-hamburg.de closing connection
```

Nachrichtenformat [RFC822]

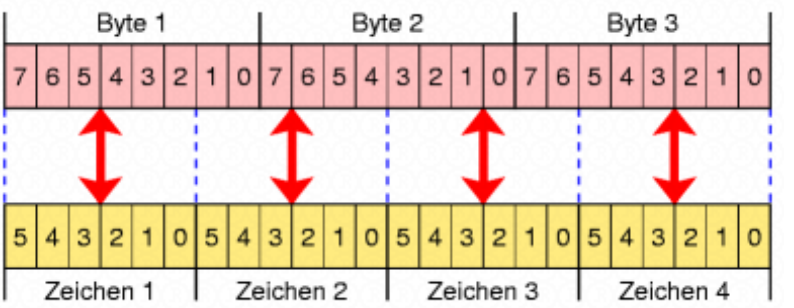
- **Header-Zeilen, z. B.**
  - From:
  - To:
  - Cc:
  - Subject:

  sind keine SMTP-Befehle!
- **Body enthält die eigentliche Nachricht**



Nachrichtencodierung

- **Alle SMTP–Nachrichten (Header und Body) müssen 7-bit ASCII-codiert sein**
  - bestimmte Zeichenfolgen sind in der Nachricht verboten (z.B. <CR><LF>.<CR><LF>).
- **Umlaute, Sonderzeichen, Word-Dokumente, Bilder, Videos, etc. müssen codiert werden!**
  - übliche Codes:
    - Base64
    - quoted printable



Multimedia-Erweiterungen (MIME)

- RFC 2045 und 2046 definieren zusätzliche Zeilen im Nachrichtenheader

MIME version

method used to encode data

multimedia data type, subtype, parameter declaration

encoded data

```
From: alice@laptop.dsl....
To: kpk@kossakowski.de
Subject: MIME-Test
MIME-Version: 1.0
Content-Transfer-Encoding:base64
Content-Type: image/jpeg

JVBERi0xLjMKJeLjz9MKNSAwIG9iago8PC9MZW5n
dGggNiAwIFIKL0ZpbHRlciAvRmxhdGVEZWNvZGUK
Pj4Kc3RyZWFtCnjaAwAAAABCmVuZHN0cmVhbQpl
bmRvYmoKNiAwIG9iago4CmVuZG9iago3IDAgb2Jq
Cjw8L1R5cGUvWE9iamVjdAovU3VidHlwZS9JbWFn
ZQovV2lkdGggODAwCi9IZWlnaHQgMjUxCi9CaXRz
UGVyQ29tcG9uZW50IDgKL0NvbG9yU3BhY2UvRGV2
```

MIME-Typen

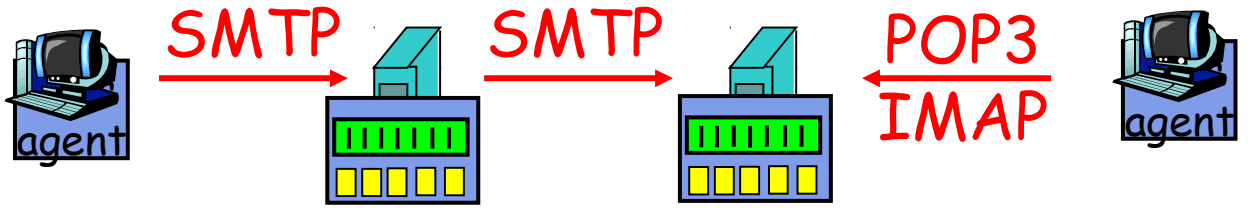
`Content-Typ: Type/Subtyp; Parameter`

- Text
  - Beispiel-Subtypen: plain, html
- Image
  - Beispiel-Subtypen: jpeg, gif
- Audio
  - Beispiel-Subtypen: basic, 32kadpcm
- Video
  - Beispiel-Subtypen: mpeg, quicktime

MIME-Typen (2)

- Application
  - Alle Daten, die in keine der übrigen Kategorien passen
- Spezielle Anwendungen
  - msword
  - octet-stream

Mailzugriffs-Protokolle



- **Abholen vom Mailserver**
  - POP: Post Office Protocol [RFC 1939]
    - Autorisierung und Download
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - größere Funktionalität und wesentlich komplexer
    - Nachrichten bleiben i. d. R. auf dem Server

POP3-Protokoll [RFC 1939]

Der User Agent (Client) holt eingegangene Nachrichten aus der User-Mailbox beim Mailserver ab.

Autorisationsphase:

- Client-Befehle:
  - USER username
  - PASS password
- Server-Antworten:
  - +OK
  - -ERR

```
S: +OK POP3 server ready
C: USER alice
S: +OK
C: PASS hungry
S: +OK
```

POP3-Protokoll [RFC 1939] (2)

Der User Agent (Client) holt eingegangene Nachrichten aus der User-Mailbox beim Mailserver ab.

Transaktionsphase:

`LIST:` Ausgabe der Nachrichtengröße
`RETR:` Nachricht übertragen
`DELE:` Nachricht löschen
`QUIT`

```
C: LIST
S: 1 498
S: 2 912
S: .
C: RETR 1
S: <message 1 contents>
S: .
C: DELE 1
C: RETR 2
S: <message 1 contents>
S: .
C: DELE 2
C: QUIT
S: +OK
```

Die Welt des Webs!

World Wide Web

Das World Wide Web wurde als universeller Informationsdienst konzipiert, um auf beliebige Ressourcen transparent zugreifen zu können. Seine Kernbestandteile sind:

- **URL / Uniform Resource Locator [RFC 2396, 3986]**

  <scheme>://<authority><path>?<query>

- **http / Hypertext Transfer Protocol [RFC 2616]**

  GET – Dokumentenabfrage vom Server

  HEAD, POST (PUT, DELETE, TRACE)

World Wide Web (2)

- **HTTP 1.0 [RFC 1945]:**

```
GET /index.html
```

- **HTTP 1.1 [RFC 2616]:**

```
GET /index.html HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/5.0
Host: www.whitehouse.gov
Accept: image/gif, image/jpeg
Accept-language:de
```

World Wide Web (3)

- Antwort des Servers nach HTTP 1.1:

```
HTTP/1.1 200 Document follows
Date: Tue, 26 Feb 2014 8:17:58 MET
Server: Apache/2.4.1
Last-modified: Mon, 17 Jun 2013 21:53:08
Content-type: text/html
Content-length: 2482
...
```

Nicht-persistente bzw. persistente Verbindungen bei HTTP

Nicht-persistent

- HTTP/1.0
- nach jeder Übertragung wird die TCP-Verbindung geschlossen!
- 2 x Round Trip Time nötig, um ein Objekt zu holen
  - Verbindungsaufbau
  - Anfrage / Antwort

Persistent

- HTTP/1.1
- Mehrere Anfragen über dieselbe TCP-Verbindung möglich!
  - (1 + n) x RTT
- ohne Pipelining: Clients warten auf Antwort
- mit Pipelining: eben kein warten

HTTP Antwort - Statuscodes

200 OK

- Anfrage war erfolgreich und die Objektdaten kommen im Datenbereich der Antwortnachricht

301 Moved Permanently

- angefragtes Objekt permanent verschoben und neue URL in der Headerzeile „location“

400 Bad Request

- Anfragenachricht nicht interpretierbar

404 Not Found

- Angeforderte Datei auf Server nicht vorhanden

505 HTTP Version Not Supported

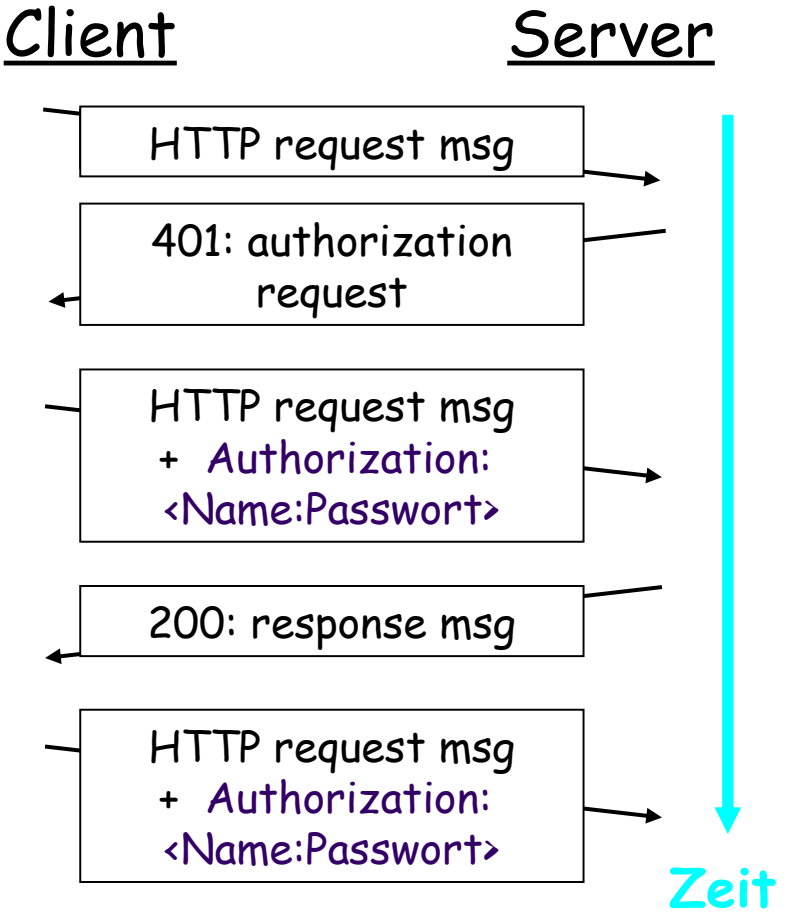
HTTP-Nachrichtenheader

Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Accept-Language	Request	The natural languages the client can handle
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Cookie	Request	Sends a previously set cookie back to the server
Date	Both	Date and time the message was sent
Upgrade	Both	The protocol the sender wants to switch to
Server	Response	Information about the server
Content-Encoding	Response	How the content is encoded (e.g., gzip)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Last-Modified	Response	Time and date the page was last changed
Location	Response	A command to the client to send its request elsewhere
Accept-Ranges	Response	The server will accept byte range requests
Set-Cookie	Response	The server wants the client to save a cookie

Benutzer-Authentifizierung: Basic Authentication

Zugriffsschutz für Serverinhalte

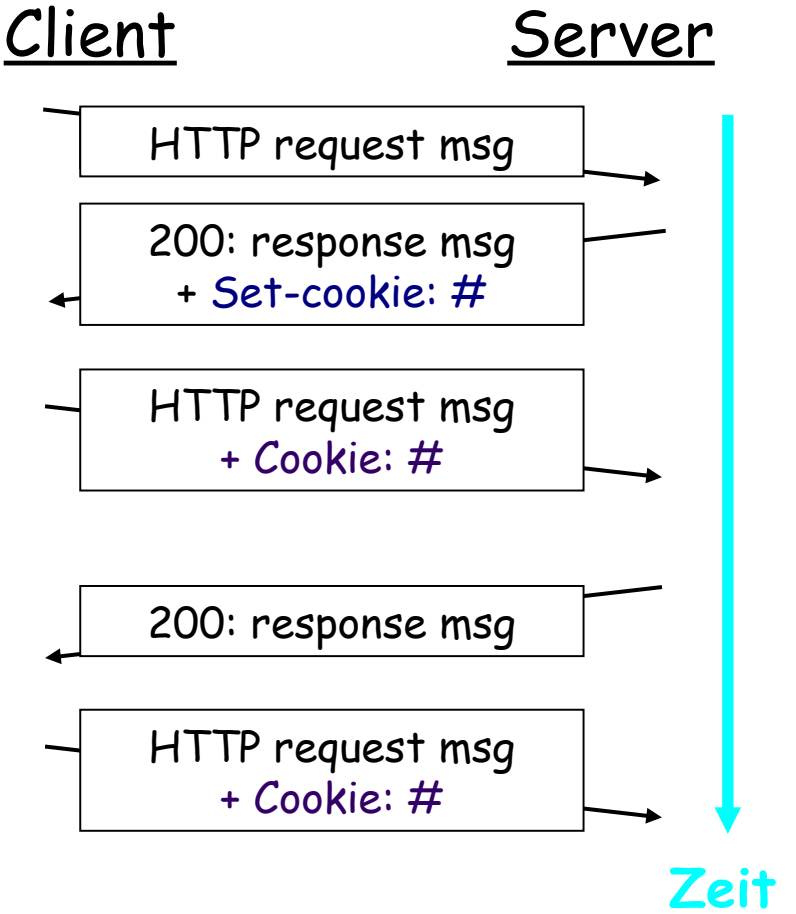
- Durch Benutzer-name und Passwort (base64-codiert)
- Zustandslos: Client muss die Autorisations-informationen in jeder Anfrage (!) angeben



Benutzer-Identifizierung: Cookies [RFC 2109]

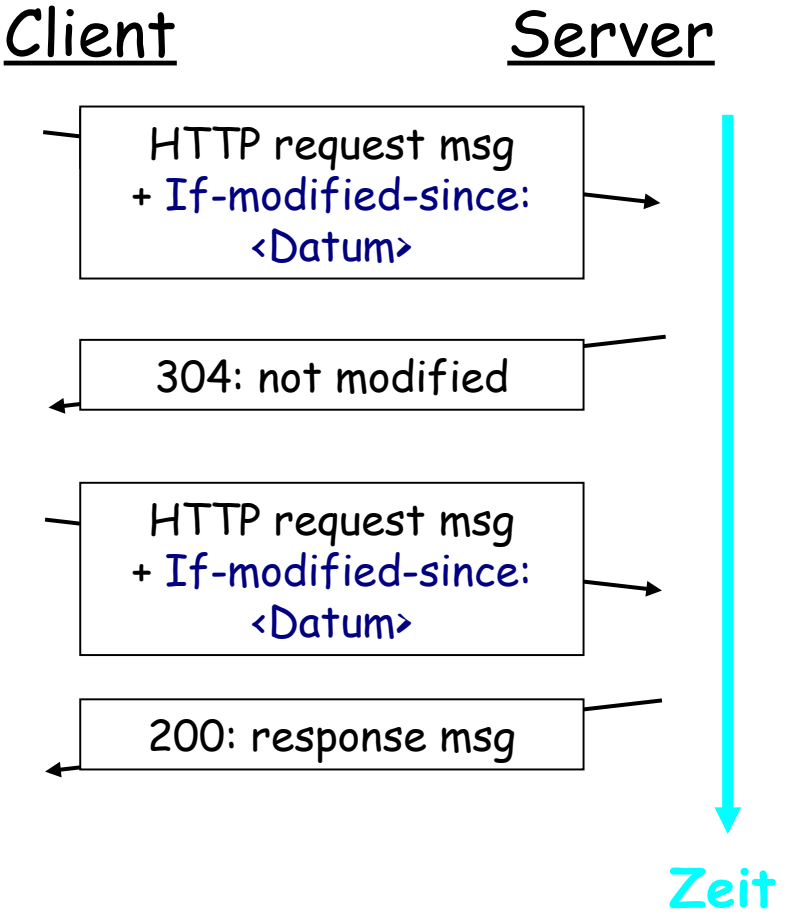
Zustandsinformationen werden auf dem Client gespeichert!

- Server sendet “Cookie” in der Antwort mit zum Clienten
- Client speichert Cookie lokal ab und liefert ihn bei neuen Abfragen mit



Caching durch den Client: Bedingter GET-Befehl (HTTP/1.1)

- **Seiten bleiben auf Clients gespeichert!**
  - **Client gibt Datum der Kopie in der Anfrage mit an**
  - **Antwort enthält kein Web-Objekt, wenn die Kopie aktuell ist**



Sicherheitsprobleme im WWW

- **HTTP / HTML / Server:**
  - Unverschlüsselte Übertragung
  - Keine Authentisierung / Authentizität
  - Cookies
  - Mobiler Code / Skripte auf Clients
  - Web-Server-Zugriff / Server-Skripte
- **Beispiele:**
  - Cross-Site-Scripting (XSS)
  - Cross-Site-Request-Forgery (CSRF)
  - Webserver-Spoofing

Beispiel:
Cross-Site-Scripting (XSS)

- "Normaler" Link:

```
<a href="http://www/default.html">
Click Here </a>
```

Dateiname

- "Böser" Link – auf gleiche Webseite:

```
<a href="http://www/<script>alert('TEST');</script>">
Click Here </a>
```

Dateiname

Beispiel:
Cross-Site-Scripting (XSS)

- "Böser" Link – auf gleiche Webseite:

```
<a href="http://www/<script>alert('TEST');</script>">
Click Here </a>
```

Dateiname

- Ergebnis = Ausgabe des Webservers:

```
<HTML>
404 page not found:
<script>alert('TEST');</script>
</HTML>
```

Dateiname

Code wird ausgeführt, wenn die Metazeichen '<' und '>' nicht vom Web-Server entfernt wurden!

Protokolle, die

die Welt nicht mehr braucht!

Protokolle, die die Welt nicht mehr brauchen sollte!

Telnet

- **Kunstwort aus Telecommunication Network**
- **Zugriff auf einen entfernten Rechner oder Netzwerkkomponente, unabhängig vom Hersteller und Betriebssystem**
  - Unterstützung der Übertragungseigen-schaften und Zeichensätze der Endgeräte
- **Der Zugriff erfolgt über den Telnet-Client auf der Kommandozeile**
- **Verbindung zum Telnet-Server: 23/tcp**

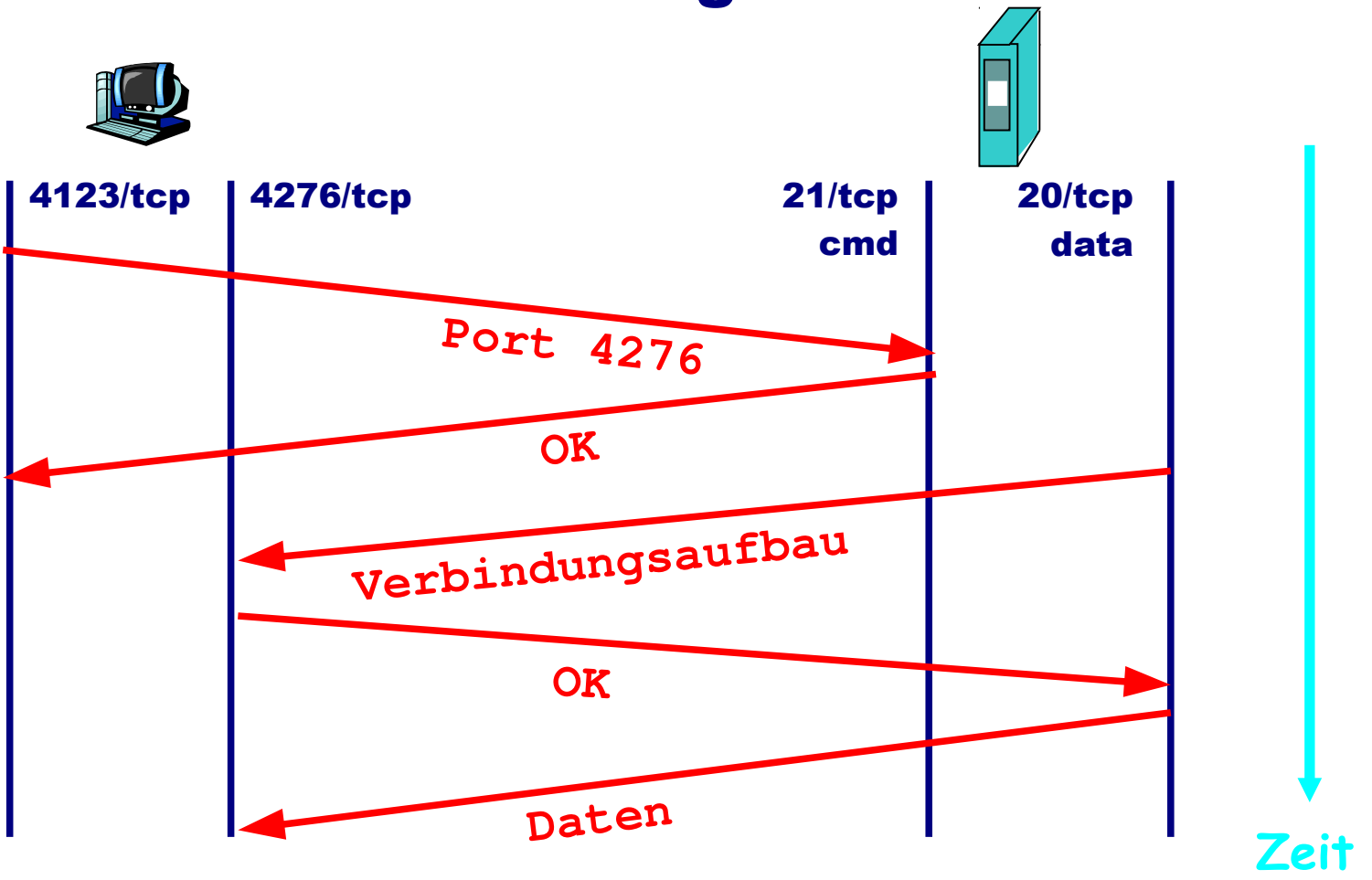
File Transfer Protocol (FTP)

- **Ermöglicht den Transfer von Dateien von bzw. zu einem entfernten Rechner**
  - [RFC 959, 3659, 5797]
- **Auch hier ein Client-/Server Modell, aber unterschiedliche Optionen**
  - Verbindung 1 / Befehle – Client/Server
  - Verbindung 2 / Daten
    - Server/Client = Aktives FTP
    - Client/Server = Passives FTP
- **Verbindung zum FTP-Server: 21/tcp**

File Transfer Protocol (FTP)

- **Aktive Steuerung der Datenverbindung**
  - Client übermittelt dem Server an 21/tcp seine Portadresse, an der er Daten empfangen möchte, mit dem Portbefehl
  - Server baut vom Standard-FTP-Port 20 eine weitere TCP-Verbindung zu diesem Client-Port auf
- **jede weitere Datenübertragung erfordert erneuten Verbindungsaufbau mit neuem Port**
  - Anforderung über den bestehenden Kanal

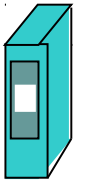
Aktiver FTP-Transfer:
Server baut Verbindung auf

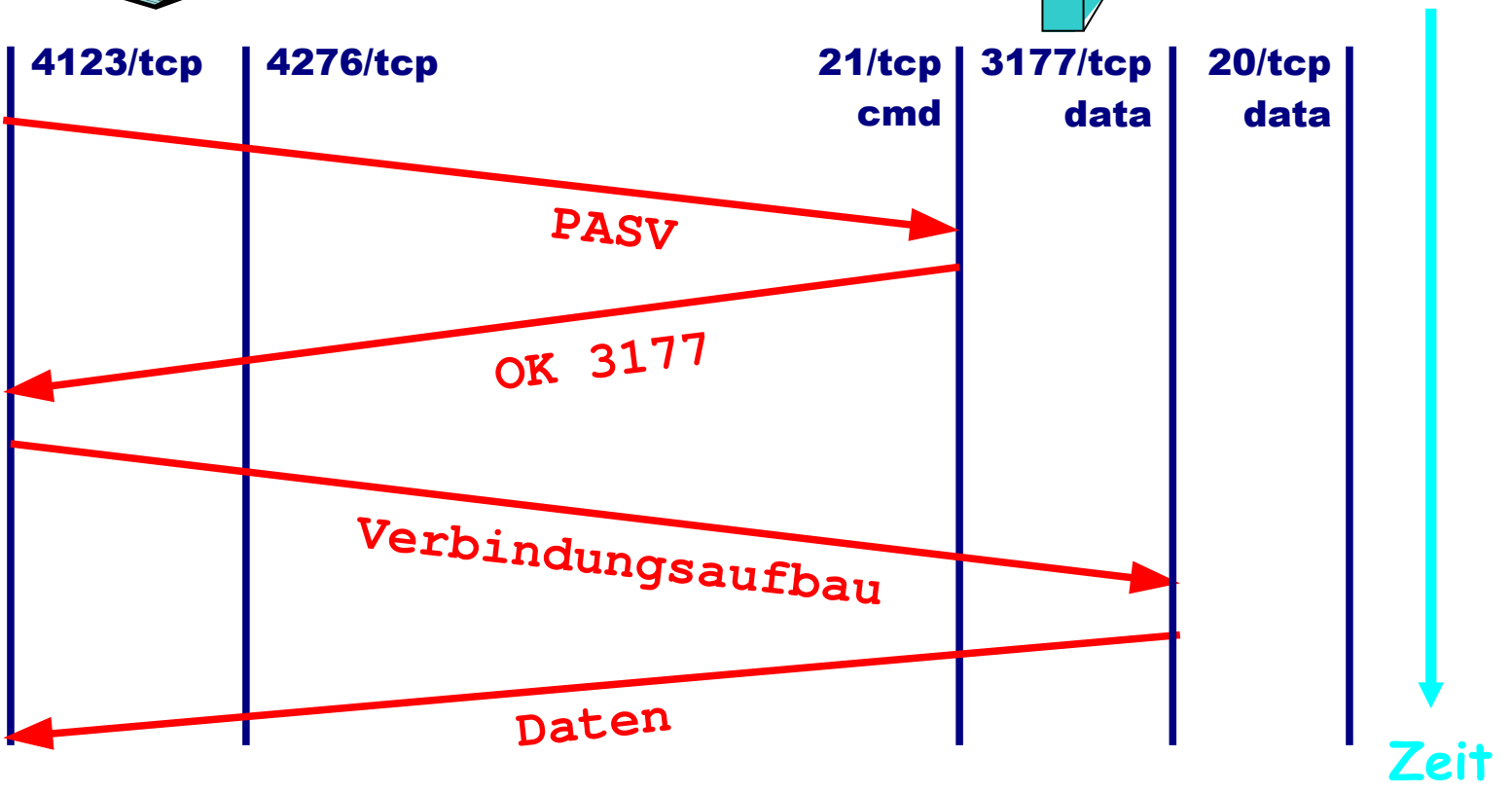


File Transfer Protocol (FTP, 2)

- **Passive Variante:**
  - Client übermittelt anstelle eines Portbefehls den PASV-Befehl
  - Server antwortet mit einer Portnummer
  - Client baut eine weitere TCP-Verbindung zu dem angegebenen Port auf und erhält angeforderte Datei

Passiver FTP-Transfer: Client baut Verbindung auf





Kontakt

Dr. Christian Keil

E-Mail: christian.keil@haw-hamburg.de