Rechnernetze
Kapitel 7: Application Layer

Prof. Dr. Wolfgang Mühlbauer
Fakultät für Informatik
wolfgang.muehlbauer@th-rosenheim.de

Wintersemester 2021/22

Slides are based on:
J. Kurose, K. Ross: Computer Networks – A Top-Down Approach
A. Tanenbaum, D. Wetherall: Computer Networks

Application Layer

- Typische **Aufgaben**
  - **Typen** der ausgetauschten Nachrichten
    - Beispiel: Anfrage („Request“) und Antwort („Response“)
  - **Syntax** der Nachrichten
    - Welche Nachrichtenfelder gibt es und wie sind diese in der Nachricht angeordnet?
  - **Semantik** der Nachrichten
    - Bedeutung der Information in den einzelnen Feldern
  - **Regeln** wann/ wie Prozesse Nachrichten senden bzw. auf diese antworten
    - Zustandsautomat

- **Offene Protokolle**
  - Definiert in "Request for Comments" (RFCs)
  - Beispiele: HTTP, SMTP

- **Proprietäre Protokolle**
  - Beispiel: Skype

Sicherheit

- ***TCP und UDP***
  - Keine Verschlüsselung
  - Passwörter, die als Klartext über Socket gesendet werden, sind überall im Internet sichtbar.

- ***TLS / SSL***
  - Verschlüsselung für TCP Verbindungen, verfügbar für alle TCP Anwendungen
  - Datenintegrität
  - Authentifizierung der Endpunkte
  - Prominentes Beispiel: HTTPS
  - TLS/SSL ist aus Rechnernetze-Sicht ein Protokoll der Anwendungsschicht

Inhalt

- **DNS**
- Web und HTTP
- E-Mail

Domain Name System: Aufgaben

- Übersetzung Hostname → IP Adresse
  - Name: `th-rosenheim.de` (einfach zu merken)
  - IP Adresse: `141.60.160.196` (lesbar für Maschinen)

- Weitere Aufgaben
  - **Host Aliasing:** Host kann mehrere Namen haben, Übersetzung
    - *Canonical* Name: `relay1.west-coast.enterprise.com`
    - *Alias* Name: `www.enterprise.com`
  - **Mailserver Aliasing**
    - Finde Mailserver für eine Domain.
    - *MX Record*: Speichert *Canonical* Name des Mailservers.
  - **Load Balancing**
    - Replizierte Webserver: Viele IP Adressen haben gleichen Namen
    - Antwort des DNS Servers bestimmt, welcher physikalische Server verwendet wird.

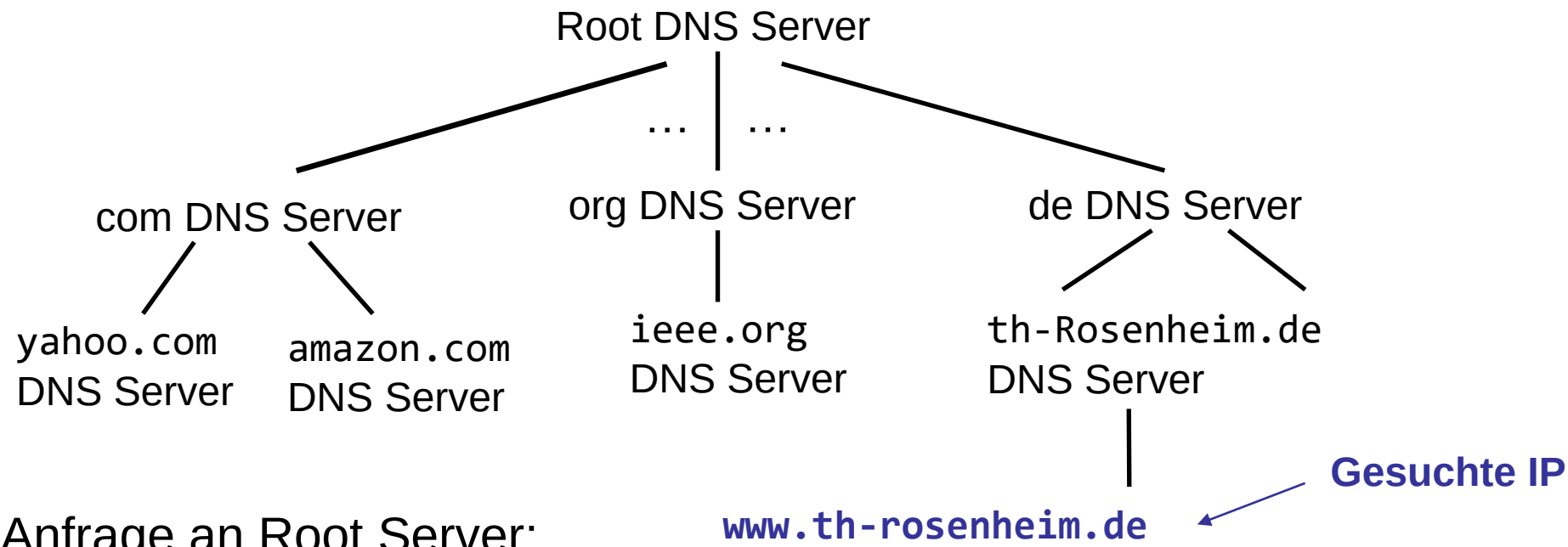
Aufbau von DNS

- **Verteiltes Verzeichnis**
  - Hierarchie von Name Servern

- **Protokoll der Application Layer**
  - Hosts und DNS Server kommunizieren miteinander.
  - Wichtige Internetfunktion wurde in der Application Layer implementiert.
  - Prinzip: „Komplexität am Rande des Internets“

- Warum **kein zentralisiertes** DNS?
  - Single Point of Failure
  - Zu hohes Verkehrsvolumen.
  - Name Server eventuell sehr weit von anfragendem Host entfernt → hohe Round Trip Time für DNS Anfragen

DNS: Durchlaufen der Hierarchie



(1) Anfrage an Root Server:
DNS Server von `.de`?

(2) Anfrage an `.de` DNS Server:
DNS Server der TH Rosenheim?

(3) Anfrage an DNS Server der TH Rosenheim:
IP Adresse des Hosts `www` in der Domain `th-rosenheim.de`?

DNS Caching

- ***Caching***
  - Lernt Nameserver eine Record, wird der Inhalt zwischengespeichert.
  - IP Adressen der TLD Server sind so gut wie immer im Cache des Resolvers. IP Adressen der Root Server müssen bekannt sein.

- ***Veralten*** von Cache-Einträgen
  - Timeout für Cache-Einträge: TTL können **veraltet sein**
  - Es dauert etwas bis sich die Änderungen im DNS verbreiten.

- Änderung von DNS Einträgen (DNS Update)
  - RFC 2136

Klassifizierung der Nameserver

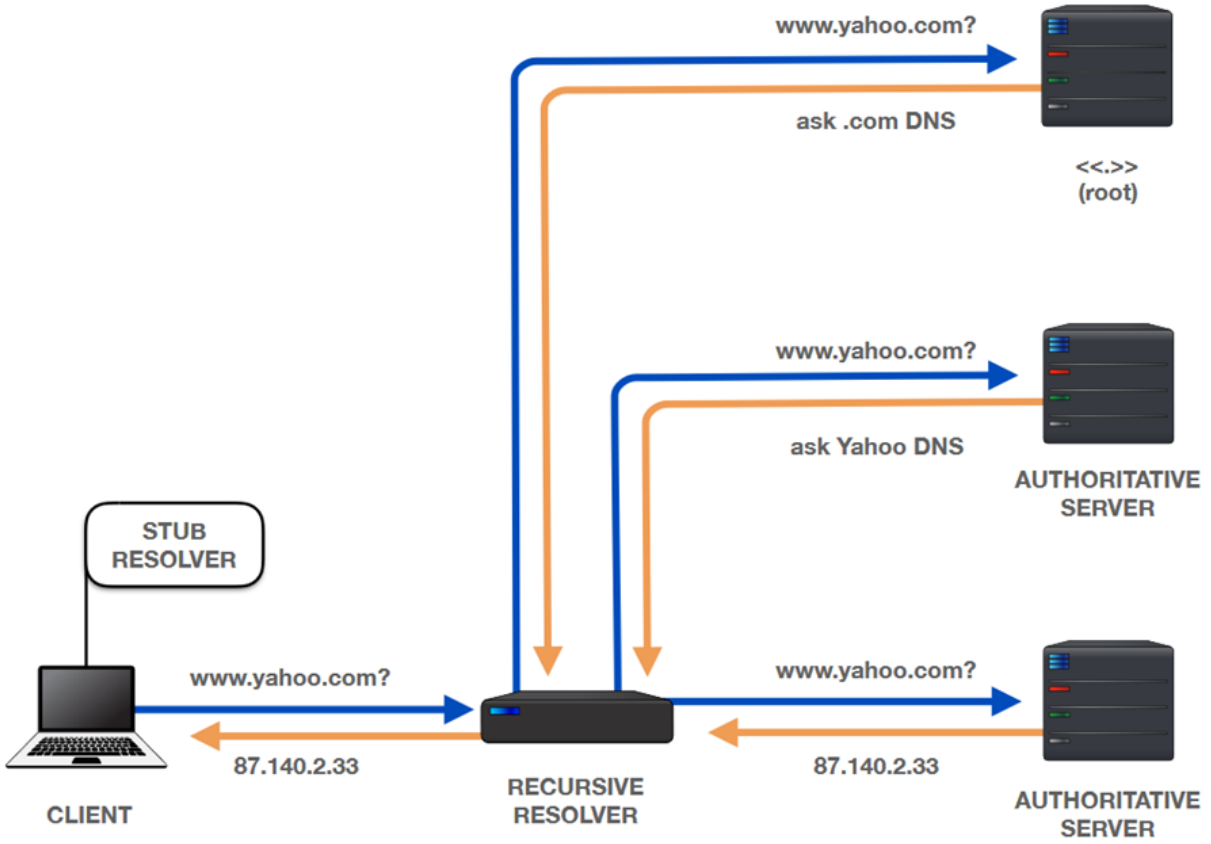
- **Root**
  - Kennt IPs aller TLD-Nameserver
  - `.de, .org, .net, .com, .edu`
- **Top-Level Domain (TLD)**
  - Beispiel: `.de` TLD Server kennt für jede `.de` Domain (hier: `th-rosenheim.de`) den zuständigen Nameserver
- **Authoritative**
  - Zuständig für IP Adresse eines Hosts.
  - Beispiel: Host `www` in der Domain `th-rosenheim.de`
- **Resolver**
  - Stellt Anfragen im Namen von Hosts („Proxy“)
  - Speichert keine verbindliche Info, nur Caching!
  - Häufig vom ISP bereitgestellt.

Domain	Intended use	Start date	Restricted?
com	Commercial	1985	No
edu	Educational institutions	1985	Yes
gov	Government	1985	Yes
int	International organizations	1988	Yes
mil	Military	1985	Yes
net	Network providers	1985	No
org	Non-profit organizations	1985	No
aero	Air transport	2001	Yes
biz	Businesses	2001	No
coop	Cooperatives	2001	Yes
info	Informational	2002	No
museum	Museums	2002	Yes
name	People	2002	No
pro	Professionals	2002	Yes
cat	Catalan	2005	Yes
jobs	Employment	2005	Yes
mobi	Mobile devices	2005	Yes
tel	Contact details	2005	Yes
travel	Travel industry	2005	Yes
xxx	Sex industry	2010	No

Quelle: Tanenbaum, Computer Networks

Wie funktioniert die Namensauflösung?

- Client benötigt die IP Adresse von `www.yahoo.com`
- Client lernt IP Adresse des Resolvers in der Regel über DHCP.
- Resolver hat Antwort nicht im Cache. Resolver arbeitet **rekursiv** und befragt mehrere Nameserver bis er die Antwort hat.
- Andere Nameserver (z.B. Root, TLD) arbeiten **iterativ** und lösen nicht final auf.



Quelle:
https://www.ripe.net/support/training/material/dnssec-training-course/dnssec-slides.pdf

DNS Einträge

- Verteilte Datenbank, die verschiedene Einträge speichern kann.
- Einträge == Resource Records (RR)

RR Format: **(name, value, type, ttl)**

type=A

- **Name:** Hostname
- **Value:** IPv4 address

type=NS

- **Name**: Name, z.B., `foo.com`
- **Value:** Name des authoritative Nameservers für diese Domain

type=CNAME

- **Name:** Alias-Name
- **Value:** Canonical Name
- **Beispiel:** `www.ibm.com` ist eigentlich `servereast.backup2.ibm.com`

type=MX

- **Name:** z.B. `foo.com`
- **Value:** Name des Mail Servers für diese Domain, z.B. `mail.foo.com`

Domain Resource Records: Typen

Type	Meaning	Value
SOA	Start of authority	Parameters for this zone
A	IPv4 address of a host	32-Bit integer
AAAA	IPv6 address of a host	128-Bit integer
MX	Mail exchange	Priority, domain willing to accept email
NS	Name server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
SPF	Sender policy framework	Text encoding of mail sending policy
SRV	Service	Host that provides it
TXT	Text	Descriptive ASCII text

Quelle: Tanenbaum, Computer Networks

Zonendatei: Konfiguration von DNS

```
; Authoritative data for cs.vu.nl
cs.vu.nl.       86400   IN   SOA     star boss (9527,7200,7200,241920,86400)
cs.vu.nl.       86400   IN   MX      1 zephyr
cs.vu.nl.       86400   IN   MX      2 top
cs.vu.nl.       86400   IN   NS      star

star            86400   IN   A       130.37.56.205
zephyr          86400   IN   A       130.37.20.10
top             86400   IN   A       130.37.20.11
www             86400   IN   CNAME   star.cs.vu.nl
ftp             86400   IN   CNAME   zephyr.cs.vu.nl

flits           86400   IN   A       130.37.16.112
flits           86400   IN   A       192.31.231.165
flits           86400   IN   MX      1 flits
flits           86400   IN   MX      2 zephyr
flits           86400   IN   MX      3 top

rowboat                 IN   A       130.37.56.201
                        IN   MX      1 rowboat
                        IN   MX      2 zephyr

little-sister           IN   A       130.37.62.23

laserjet                IN   A       192.31.231.216
```

Der Host "`star`" ist der Nameserver der Domain "`cs.vu.nl`"

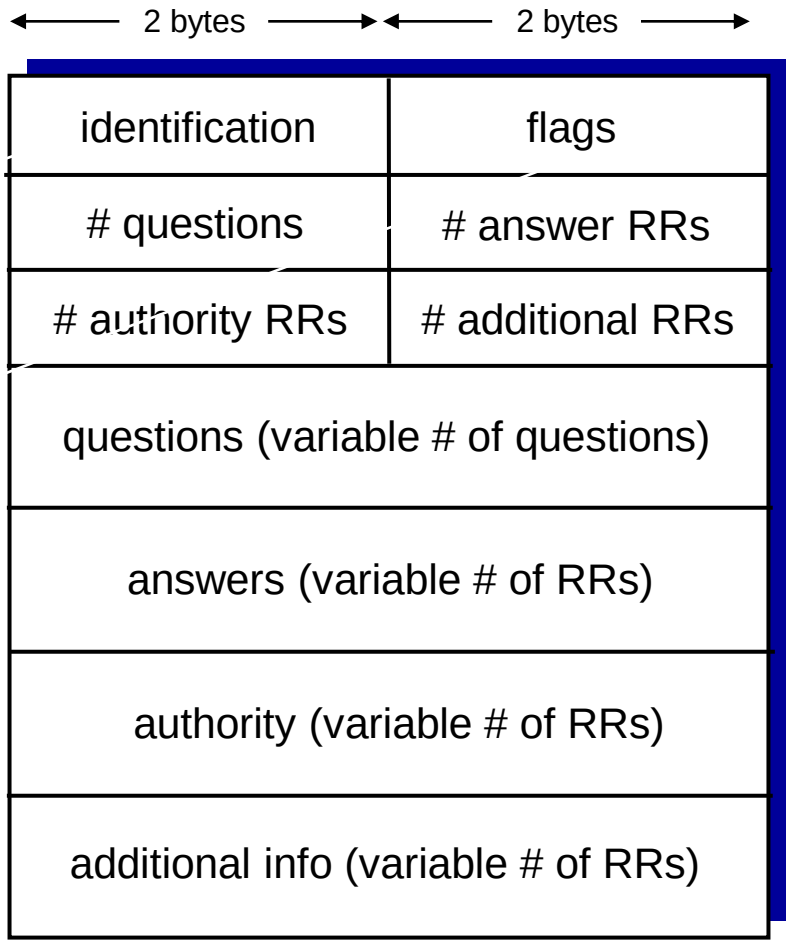
Die IP des Nameservers ist 130.37.56.205

`www` verweist auf den Host star (Alias)

Mailserver

DNS Protokoll: Aufbau der Nachrichten

- ***Query*** und ***Reply*** haben das gleiche Format
- ***Identification***
  - Identisch für Query und zugehörige Reply
- ***Flags***
  - Query oder Reply?
  - Recursive Query erwünscht
  - Recursion verfügbar
  - Antwort ist „authoritative“
- Neben der eigentlichen Antwort in „***answers***“ können ungefragt gleich weitere wichtige Infos in „***additional***“ mitgeteilt werden



Betrieb eines eigenen Nameservers / Domain

- Beispiel: Neues Startup „*NetworkTec*“

- Registriere *networktec.de*
  - Registrar für `.de` Domain: *Denic*
  - Informiere Registrar über IP Adresse des eigenen *Name*servers
    - Hier: `dns1.networktec.de`
  - Registrar fügt seinem (TLD) Namesever 2 Records hinzu

    `(networktec.de, dns1.networktec.de, NS)`

    `(dns1.networktec.de, 212.212.212.1, A)`

- Auf eigenem Nameserver
  - Lege Type A Record für www.network.tec an
  - Lege Type MX Record für network.tec an
  - …

Publikums-Joker: DNS

Welche der folgenden Aussagen ist ***falsch***?

A. DNS ist ein Protokoll der Anwendungsschicht.

B. Die Root DNS Server im Internet arbeiten rekursiv.

C. Im Standard DNS werden DNS Anfragen unverschlüsselt übertragen.

D. DNS basiert auf UDP.

Inhalt

- DNS
- **Web und HTTP**
- E-Mail

Hypertext Transfer Protokoll (HTTP)

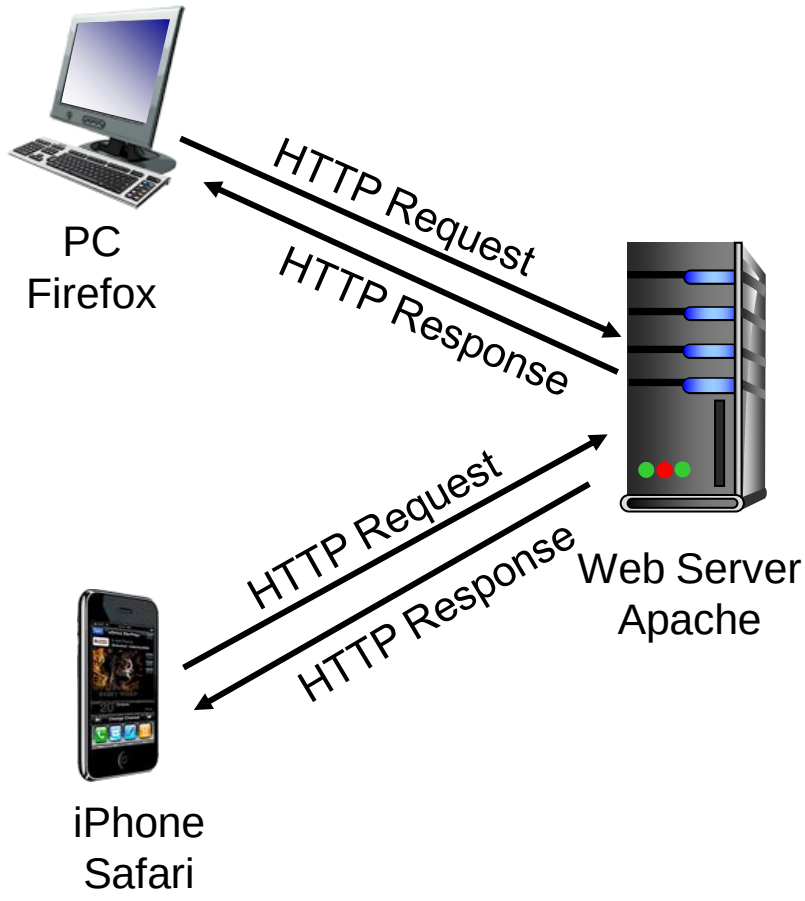
- Webseite / Webobjekt adressierbar durch **Uniform Resource Locator (URL)**
  - http://www.phdcomics.com/comics.php

- **HTTP Client**
  - Baut TCP Verbindung zu passendem Port auf, bei HTTP (fast immer) Port 80
  - Senden des HTTP Requests und Auswerten der Antwort
  - Ggfs. Nachladen weiterer URLs (z.B. Bilder), um die Seite anzuzeigen.
  - Schließen der TCP Verbindung

- **HTTP Server**
  - Akzeptieren von ankommenden TCP Verbindungen
  - Abbilden von ankommenden Anfragen auf Ressourcen (z.B. Datei)
  - Laden der Ressource und Senden an den Client.
  - Server ist „**stateless**“: Er merkt sich nichts bezüglich früherer Anfragen des Clients

Hypertext Transfer Protokoll (HTTP)

- HTTP: Application Layer Protokoll des Webs
  - Nicht verwechseln mit HTML!
  - HTTP kann HTML-Dokumente übertragen
- **Request-Response-**Protokoll auf der Basis von TCP
- Client-Server Prinzip
  - **Client**: Browser, der Webseiten anfordert („Request), empfängt und die Web-Objekte darstellt
  - **Server**: Web Server sendet über HTTP die angefragten Web-Objekte



HTTP Request Nachricht

- Verschiedene Request-Typen
- ASCII, Textformat

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

Request Zeile
(GET, POST,
HEAD commands)

Header-Zeilen

Carriage Return,
Line Feed am Zeilen-
anfang bedeutet Ende
des Headers

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

Publikums-Joker: HTTP

Welche der folgenden Aussagen ist ***falsch***?

A. HTTP benötigt zwingend eine TCP Verbindung.

B. Der HTTP-Header ist "human-readable", kein Bytecode

C. HTTP überträgt stets HTML-Dateien.

D. HTTP arbeitet nach dem GET-RESPONSE Prinzip.

HTTP Response – Status Codes

- Status Code erscheint in der 1. Zeile der Nachricht vom Server an den Client

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n
\r\n
data data data data data ...
```

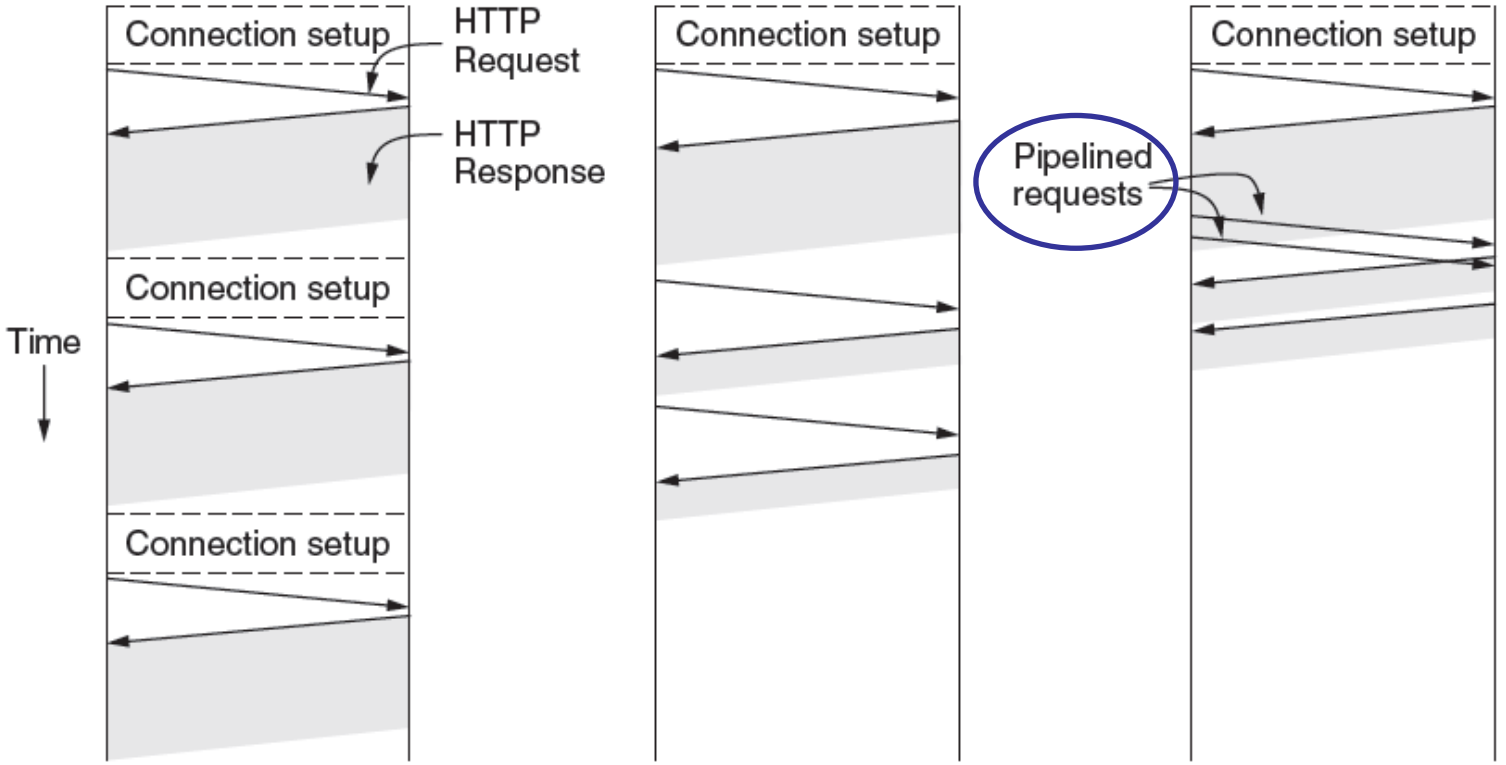
- **200 OK**
  - Request war erfolgreich
- **301 Moved Permanently**
  - Das angeforderte Objekt wurde verschoben; der neue Ort wird in der Nutzlast spezifiziert
- **400 Bad Request**
  - Die Anfrage wurde vom Server nicht verstanden
- **404 Not Found**
  - Das angeforderte Dokument wurde auf dem Server nicht gefunden
- **505 HTTP Version not supported**

HTTP Upload

- POST Methode
  - Webseiten enthalten oft HTML Formulare und Eingabefelder
  - Die Eingabe wird in der HTTP Nutzlast einer POST Nachricht übertragen.

- URL Methode
  - Verwendet eine GET Nachricht
  - Die Eingabe wird im URL Feld der Nachricht hochgeladen
  - Beispiel: www.somesite.com/animalsearch?monkeys&banana

Persistent HTTP and HTTP Pipelining



Quelle: Tanenbaum, Computer Networks

1 TCP Verbindung für jede Anfrage

Mehrere Anfragen über die gleiche TCP Verbindung

Mehrere ***parallele*** Anfragen über die gleiche TCP Verbindung

HTTP Verbesserungen

- **Non-persistent** versus **persistent** HTTP
  - *Non-persistent*
    - Höchstens 1 Webobjekt wird über 1 TCP Verbindung gesendet, Verbindung wird dann geschlossen
    - Folge: Unter Umständen mehrere TCP Verbindungen für 1 Webseite notwendig
    - 2RTTs pro Objekt, großer Overhead
  - *Persistent*
    - Mehrere Webobjekte können über gleiche TCP Verbindung gesendet werden
    - Default-Einstellung in den meisten Webbrowsern

- **Pipelining**
  - Webbrowser stellen über gleiche TCP Verbindung neue Anfrage, ohne Antwort auf vorherige Anfrage abzuwarten.
  - Selten verwendet! Stattdessen eher Einsatz mehrere TCP Verbindungen.

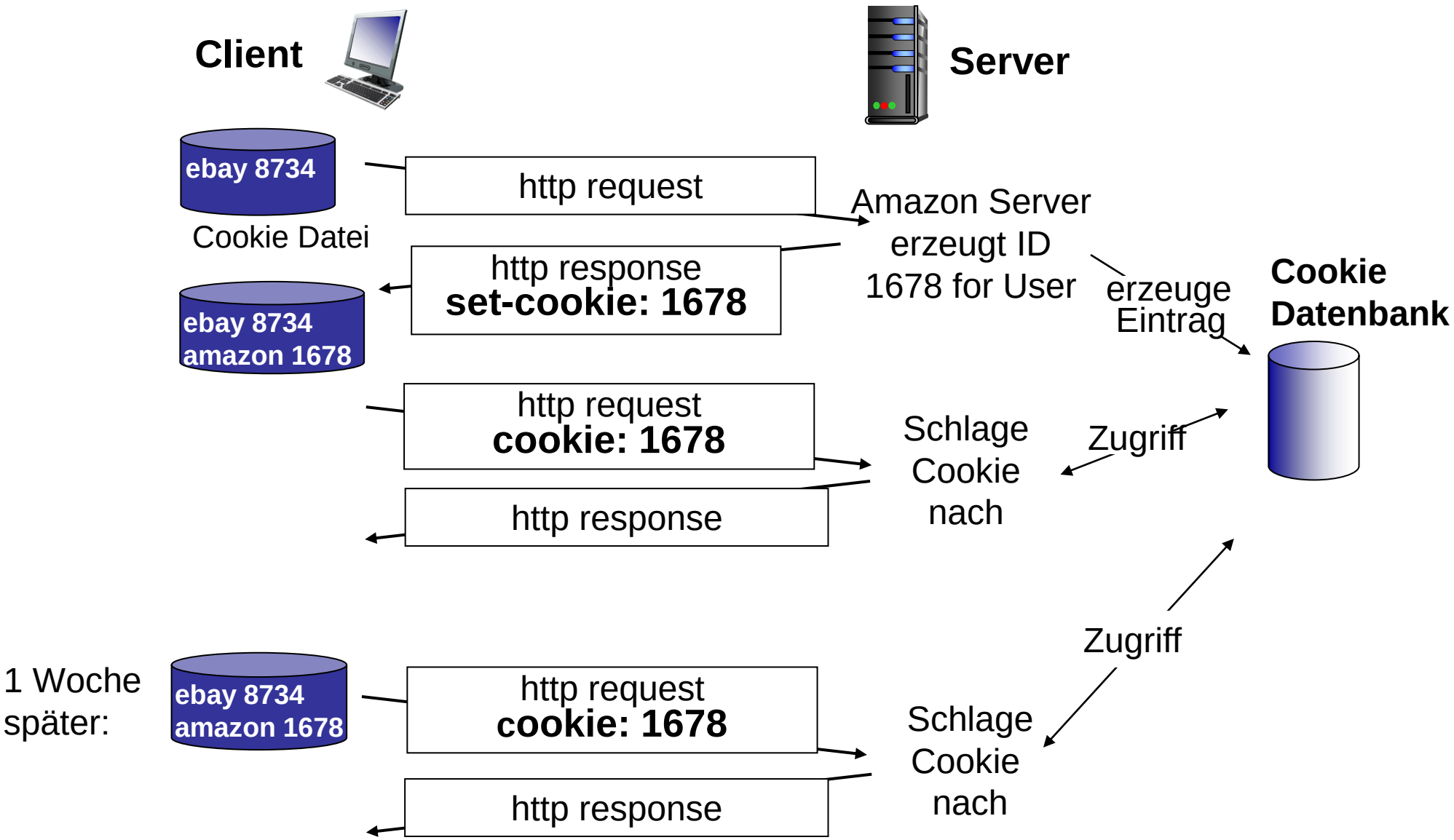
Cookies

- **Ziel**
  - HTTP ist "stateless", d. h. es vergisst sofort die letzte Anfrage.
  - Wie erkennt Webserver Benutzer beim nächsten Besuch wieder?
  - Speichern von Zustand über HTTP Sessions hinweg.

- **Einsatz von Cookies**
  - Identifikation von Web-Benutzern
  - Einkaufswagen in e-Shops, Kaufempfehlungen
  - Speichern von Session State z.B. bei Webmail

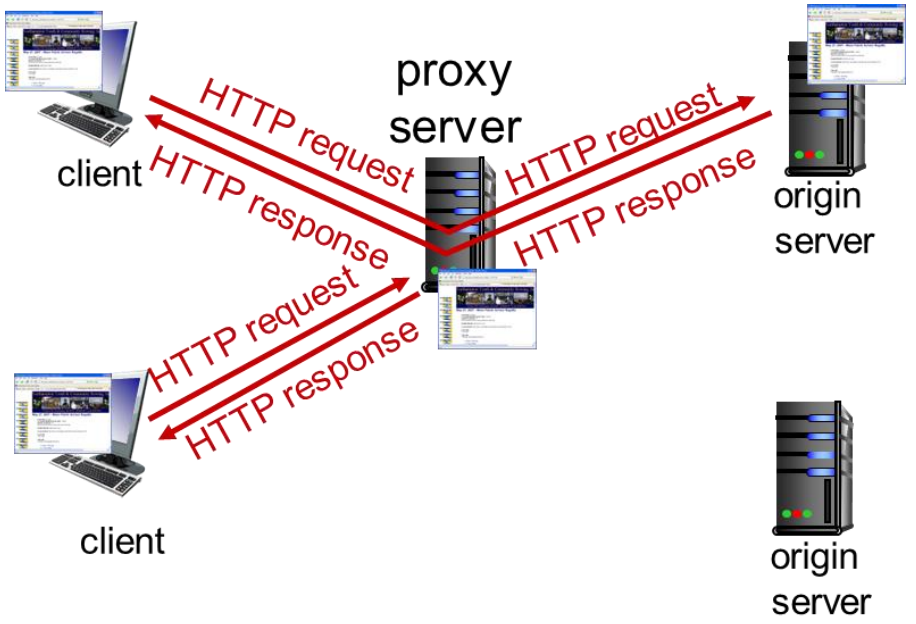
- **Cookies:** 4 Komponenten
  - Headerzeile im HTTP Response
  - Headerzeile im HTTP Request
  - Datei auf dem Computer des Nutzers, verwaltet durch Browser
  - Cookie Datenbank auf Webserver

Cookies


Client
Server
ebay 8734
Cookie Datei
http request
Amazon Server
erzeugt ID
1678 for User
erzeuge
Eintrag
Cookie
Datenbank
http response
set-cookie: 1678
ebay 8734
amazon 1678
http request
cookie: 1678
Schlage
Cookie
nach
Zugriff
http response
Zugriff
1 Woche
später:
ebay 8734
amazon 1678
http request
cookie: 1678
Schlage
Cookie
nach
http response

Web Caching

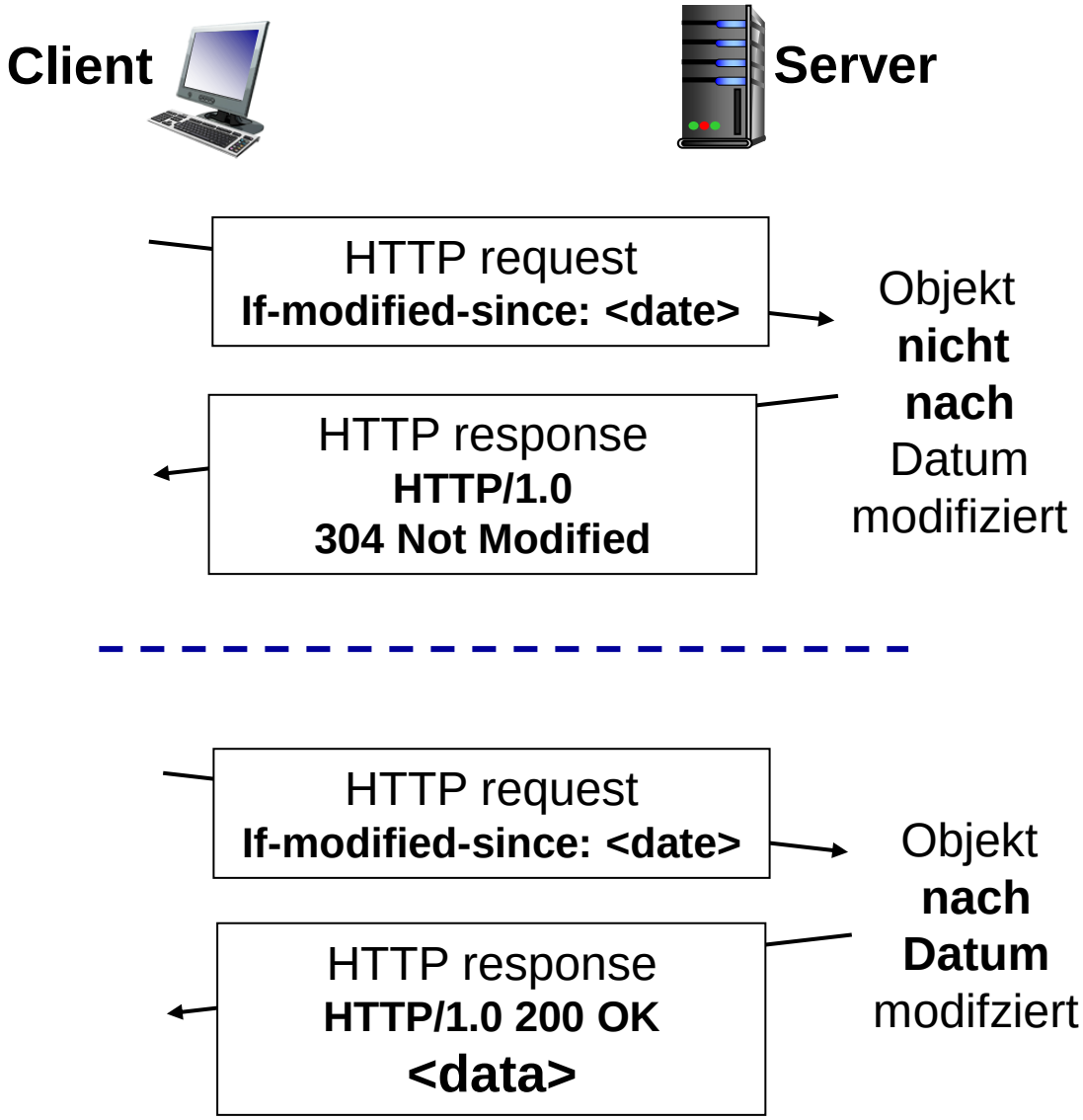
- **Ziele**
  - Kürzere Antwortzeiten für einen Web Request
  - Einsparen von Traffic auf Link zu Provider



- Konfiguration eines Proxy Server im Web Browser
- Browser sendet alle Anfragen an einen Proxy/Cache
  - Falls Objekt in Cache: Cache liefert Objekt zurück
  - Andernfalls: Proxy stellt Request an Original-Server, speichert Objekt in Cache und beantwortet Anfrage

Web Caching: Bedingtes GET

- **Ziel**
  - Server liefert Objekt nur aus, falls Cache eine ältere Version hat.
- **Client**
  - Spezifiziert Datum der zwischengespeicherten Kopie im HTTP Request
- **Server**
  - Antwort enthält kein Objekt falls zwischengespeicherte Kopie aktuell ist.



Webseiten

- Webseite besteht aus mehreren **Objekten,** z.B.
  - HTML Datei
  - JPEG Image
  - Audio Datei
  - Javascript
  - …
- Meist zentraler Einstieg in Webseite, z.B.
  - `index.html` referenziert nachzuladende Objekte der Webseite.
  - `php.ini:` Falls Webseite PHP Code enthält.
- **MIME** gibt Information über den Typ des Inhalts
  - Beispiel: `text/html, image/jpeg, audio/mpeg`
  - Bei MIME-Type `text/html`: Direktes Anzeigen der Seite durch Browser
- Unterscheidung zwischen
  - **statischen** Webseiten, die für jeden Benutzer gleich aussehen.
  - **dynamischen** Webseiten, die bei jedem Aufruf entweder vom Client oder vom Server automatisch generiert werden.

Statische Webseiten / HTML Versionen

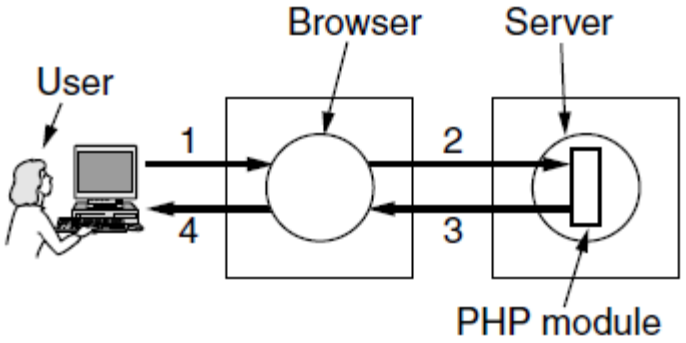
Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	x	x	x	x	x
Images	x	x	x	x	x
Lists	x	x	x	x	x
Active maps & images		x	x	x	x
Forms		x	x	x	x
Equations			x	x	x
Toolbars			x	x	x
Tables			x	x	x
Accessibility features				x	x
Object embedding				x	x
Style sheets				x	x
Scripting				x	x
Video and audio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

Quelle: Tanenbaum, Computer Networks

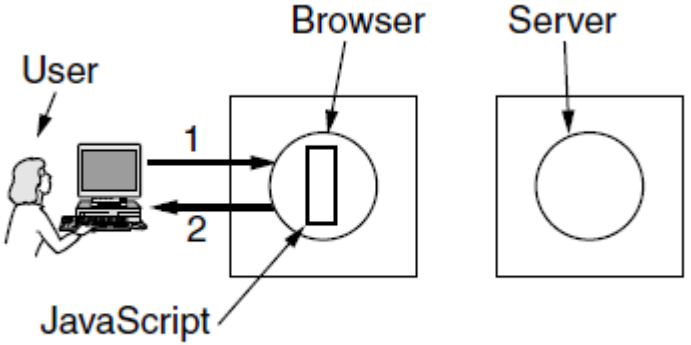
Dynamische Webseiten

- Sehen für jeden Benutzer anders aus.

- Können durch Client oder durch Server generiert werden.
  - Server-Side: PHP, CGI, usw. erlauben es z.B. durch "HTML Forms" übertragene Parameter auf dem Server auszuwerten und dann die HTML Seite zu erzeugen und zurückzuliefern.
  - Client-Side: JavaScript, VBScript. Innerhalb von <script> kann in eine HTML Seite Code eingebettet werden, der dann im Browser ausgeführt



Server-side scripting mit PHP

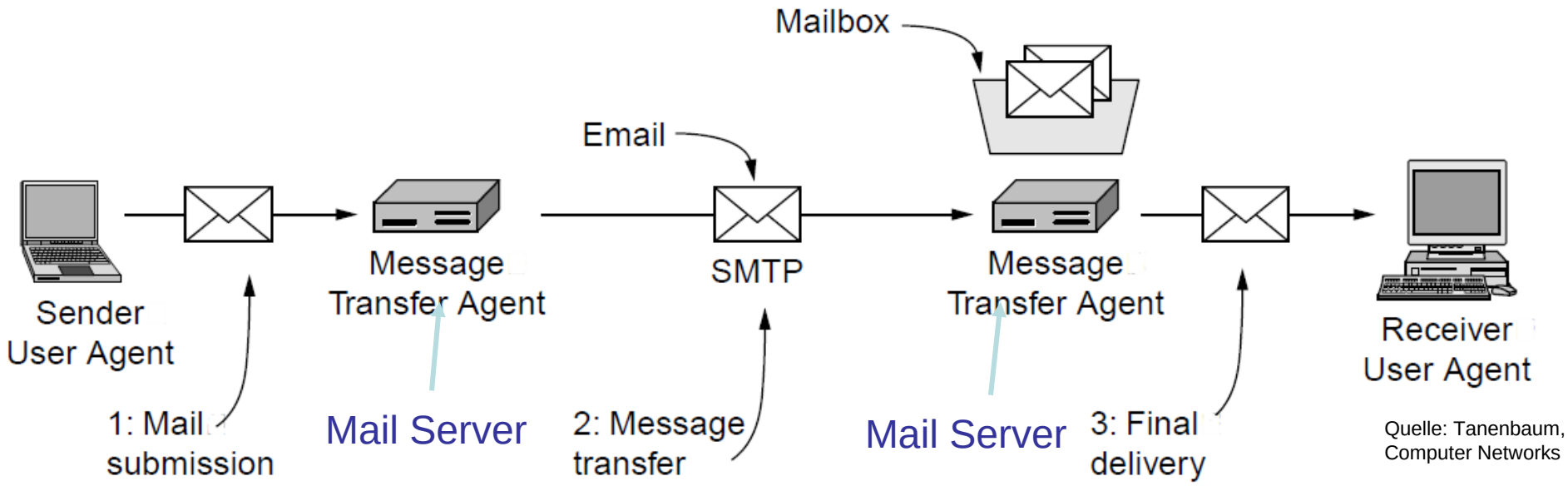


Client-side Scripting mit JavaScript

Inhalt

- DNS
- Web und HTTP
- **E-Mail**

E-Mail Infrastruktur



Quelle: Tanenbaum, Computer Networks

- **User Agent**
  - Mailprogramm (z.B. Thunderbird)
  - Sender User Agent und Receiver User Agent.
  - Kommunikation mit dem Mail Server

- **Mail Server / Message Transfer Agent**:
  - Zwischenspeichern eingehender Nachrichten bis der Benutzer sie abruft.
  - Zwischenspeichern ausgehender Nachrichten bis Zustellung über SMTP möglich.
  - SMTP: Protokoll zwischen Mail Servern.

SMTP: Simple Mail Transfer Protokoll

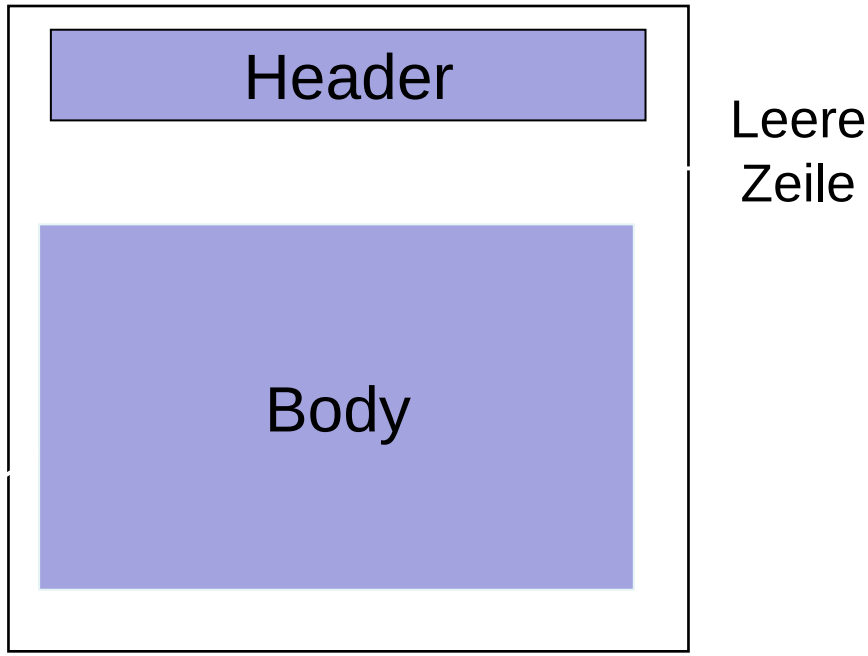
- Verwendet TCP
  - Server wartet auf Port 25
- Direkter Transfer
  - Der sendende Mail Server überträgt Daten direkt zum empfangenden Mail Server ohne Verwendung von Zwischenstationen.
- 3 Phasen der Kommunikation
  - Verbindungsaufbau: „Handshaking“ auf Schicht 5!
  - Übertragung der Nachrichten: 7-Bit ASCII
  - Schließen der Verbindung.
- Vergleich mit HTTP
  - “Request/Response“ ähnlich wie bei HTTP
  - HTTP: Pull-Verfahren, SMTP: Push-Verfahren
  - HTTP: 1 Objekt pro Antwort, SMTP: Mehrere Objekte in Multipart-Nachricht

Beispiel einer SMTP Interaktion


```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Nachrichtenformat einer E-Mail, RFC 822

- Nicht verwechseln!
  - E-Mail Protokoll: SMTP, RFC 821
  - E-Mail Nachrichtenformat, RFC 822



- Nachrichtenformat:
  - Definiert Metadaten einer E-Mail
  - Beispiele: From, Subject
  - Nicht verwechseln mit SMTP Keywords FROM, RCPT, etc.

- Body: die „Nachricht“
  - Der eigentliche Text

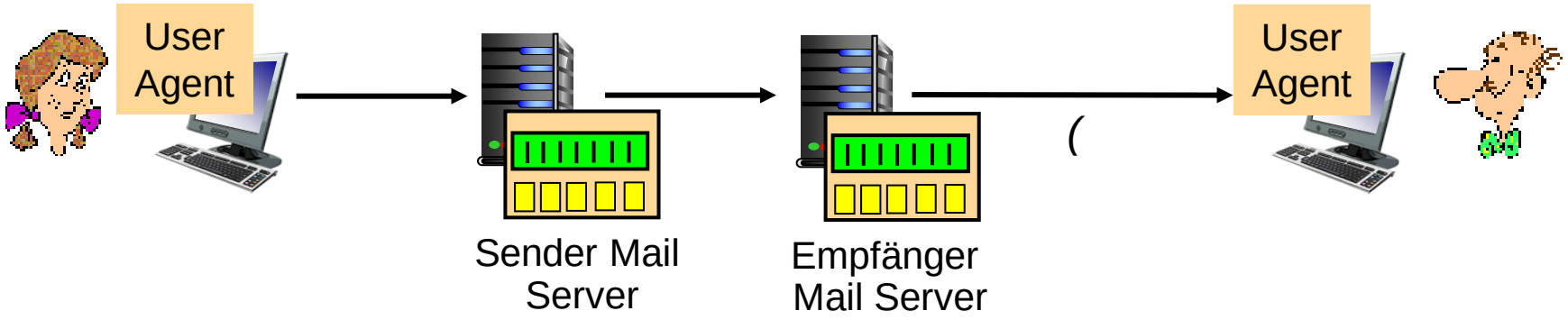
Nachrichtenformat einer E-Mail: Metadaten

Header	Meaning
To:	Email address(es) of primary recipient(s)
Cc:	Email address(es) of secondary recipient(s)
Bcc:	Email address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	Email address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

Quelle: Tanenbaum, Computer Networks

Header	Meaning
Date:	The date and time the message was sent
Reply-To:	Email address to which replies should be sent
Message-Id:	Unique number for referencing this message later
In-Reply-To:	Message-Id of the message to which this is a reply
References:	Other relevant Message-Ids
Keywords:	User-chosen keywords
Subject:	Short summary of the message for the one-line display

Mailprotokolle / Zusammenfassung



- SMTP: Liefern der E-Mail bis zum Empfänger Mail Server
  - Warum 2-Stufen-Verfahren?
- Mail Access Protokolle: Abrufen („Pull“) vom Mail Server
  - SMTP kann nicht verwendet werden, da es „Push“-basiert ist
  - Lösung: Eigene Protokolle
    - POP: Post Office Protocol
    - IMAP: Internet Mail Access Protocol
    - HTTP: Webmail Dienste, z.B. Gmail, Hotmail, etc.