

Kapitel 2

Anwendungsschicht

Ein Hinweis an die Benutzer dieses Foliensatzes:

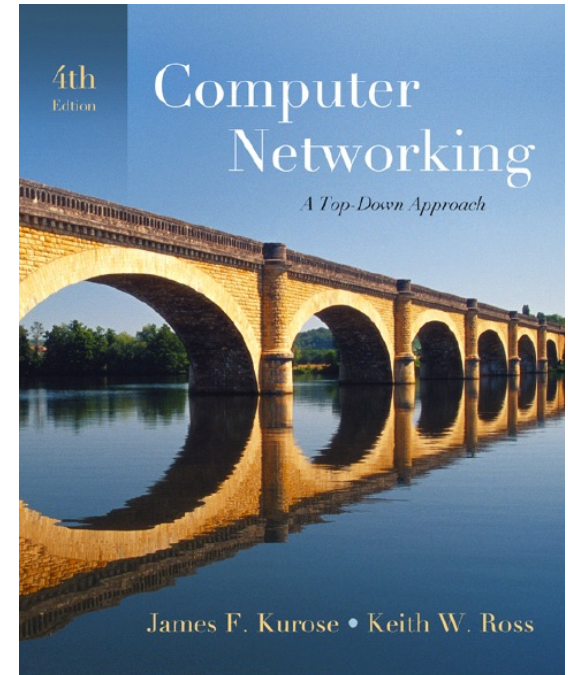
Wir stellen diese Folien allen Interessierten (Dozenten, Studenten, Lesern) frei zur Verfügung. Da sie im PowerPoint-Format vorliegen, können Sie sie beliebig an Ihre Bedürfnisse anpassen. Wir haben sehr viel Arbeit in diesen Foliensatz investiert. Als Gegenleistung für dessen Verwendung bitten wir Sie um Folgendes:

- Wenn Sie diese Folien (z.B. in einer Vorlesung) verwenden, dann nennen Sie bitte die Quelle (wir wollen ja, dass möglichst viele Menschen unser Buch lesen!).
- Wenn Sie diese Folien auf einer Webseite zum Herunterladen anbieten, dann geben Sie bitte die Quelle und unser Copyright an diesem Foliensatz an.

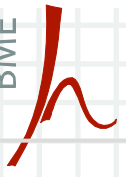
Danke und viel Spaß beim Lehren und Lernen mit diesem Foliensatz! JFK/KWR

Copyright der englischen Originalfassung 1996–2007
J.F Kurose and K.W. Ross, alle Rechte vorbehalten.

Deutsche Übersetzung 2008
M. Mauve und B. Scheuermann, alle Rechte vorbehalten.

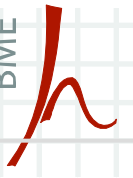


*Computernetzwerke: Der
Top-Down-Ansatz ,
4. Ausgabe.
Jim Kurose, Keith Ross
Pearson, Juli 2008.*



Kapitel 2: Anwendungsschicht

- 2.1 Grundlagen
- 2.2 Web und HTTP, HTTP/2
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS



Kapitel 2: Anwendungsschicht

Unsere Ziele:

- Konzeption und Implementierung von Protokollen der Anwendungsschicht
 - Dienstmodelle der Transportschicht
 - Client-Server-Paradigma
 - Peer-to-Peer-Paradigma
- Durch das Untersuchen konkreter Protokolle etwas Allgemeines über Protokolle lernen
 - HTTP
 - SMTP / POP3 / IMAP
 - DNS
- Kommunikation von Netzwerkanwendungen
 - Sockets

Einige Netzanwendungen

- E-Mail
- Web
- Instant Messaging
- Terminalfernzugriff
- P2P-Filesharing
- Netzwerkspele
- Streaming von Videoclips
- Voice over IP (VoIP)
- Videokonferenzen
- Cloud Computing

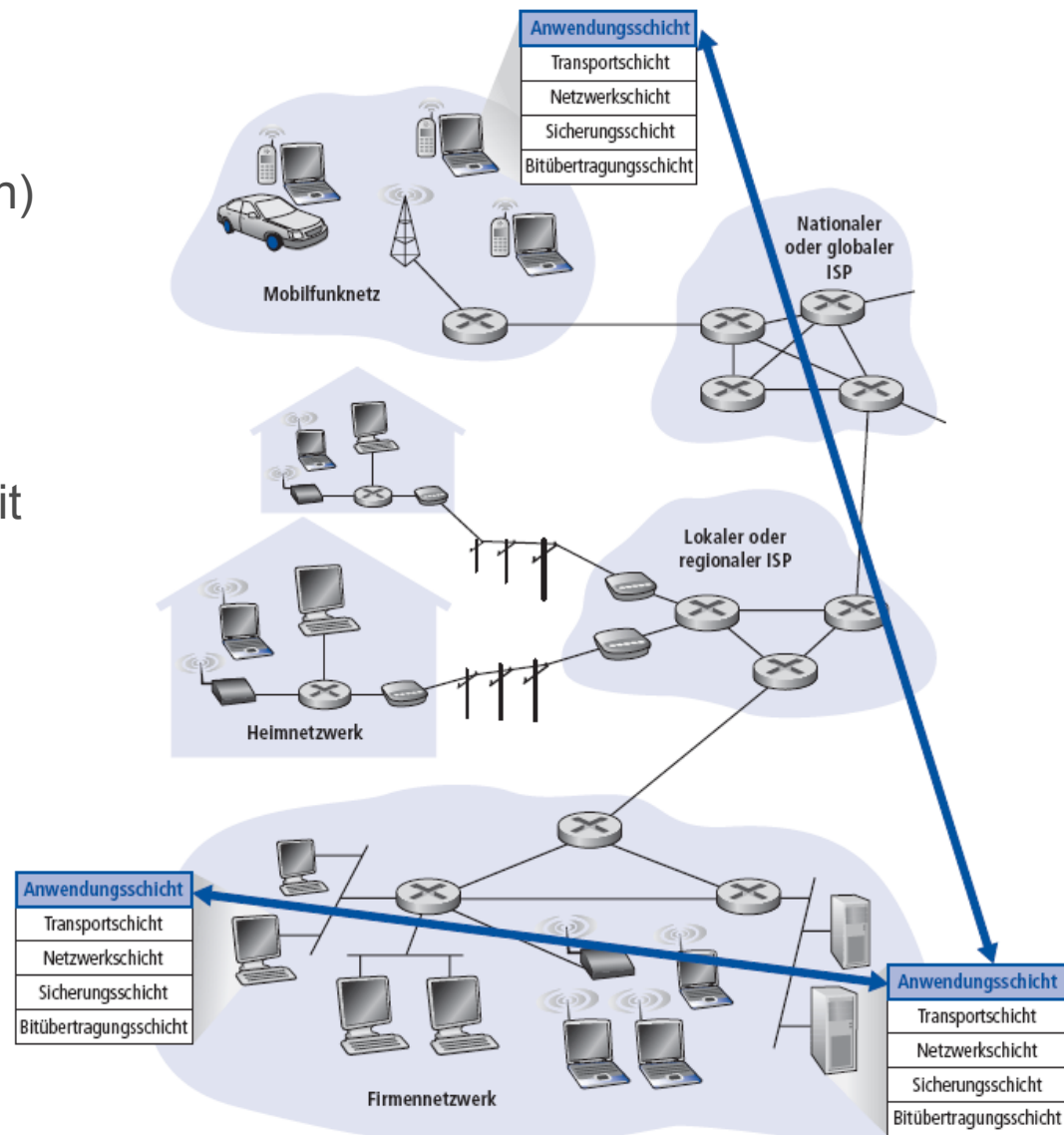
Entwickeln einer Netzwerkanwendung

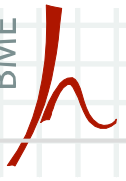
Schreibe Programme, die

- auf mehreren (verschiedenen) Endsystemen laufen
- über das Netzwerk kommunizieren
- Beispiel: Die Software eines Webserver kommuniziert mit dem Browser (Software)

Kaum Software für das Innere des Netzwerkes

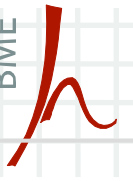
- Im Inneren des Netzwerkes werden keine Anwendungen ausgeführt
- Die Konzentration auf Endsysteme erlaubt eine schnelle Entwicklung und Verbreitung der Software





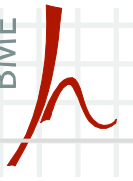
Kapitel 2: Anwendungsschicht

- 2.1 Grundlagen
- 2.2 Web und HTTP, HTTP/2
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS



Verschiedene Architekturen

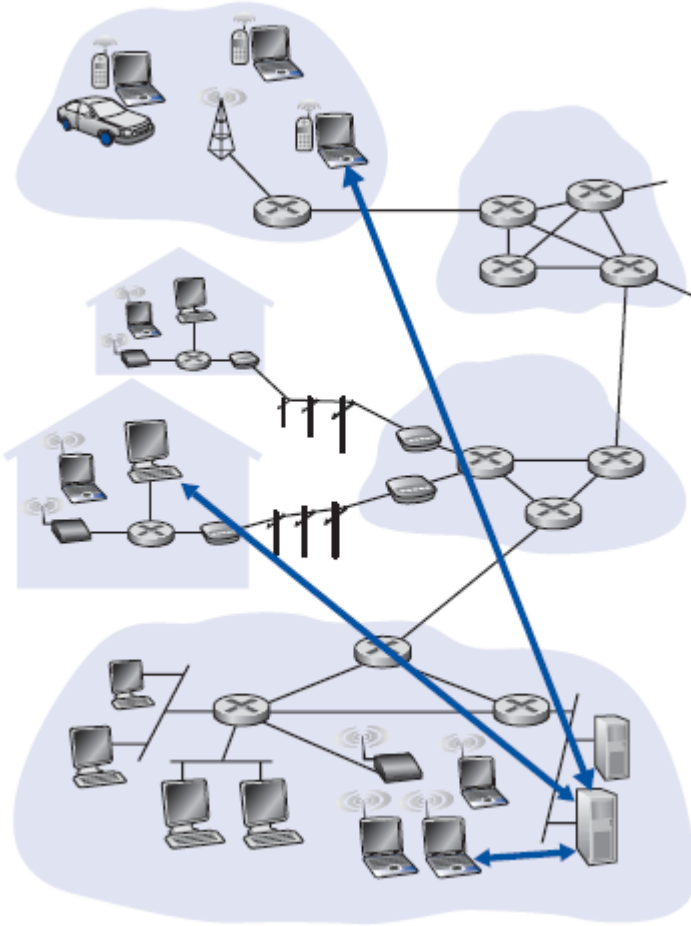
- Client-Server
- Peer-to-Peer (P2P)
- Kombination von Client-Server und P2P



Verschiedene Architekturen

- Client-Server
- Peer-to-Peer (P2P)
- Kombination von Client-Server und P2P

Client-Server-Architektur



Server:

- Immer eingeschaltet
- Feste IP-Adresse
- Serverfarmen, um zu skalieren
- Virtuelle Cloud-Backend
- Containerization

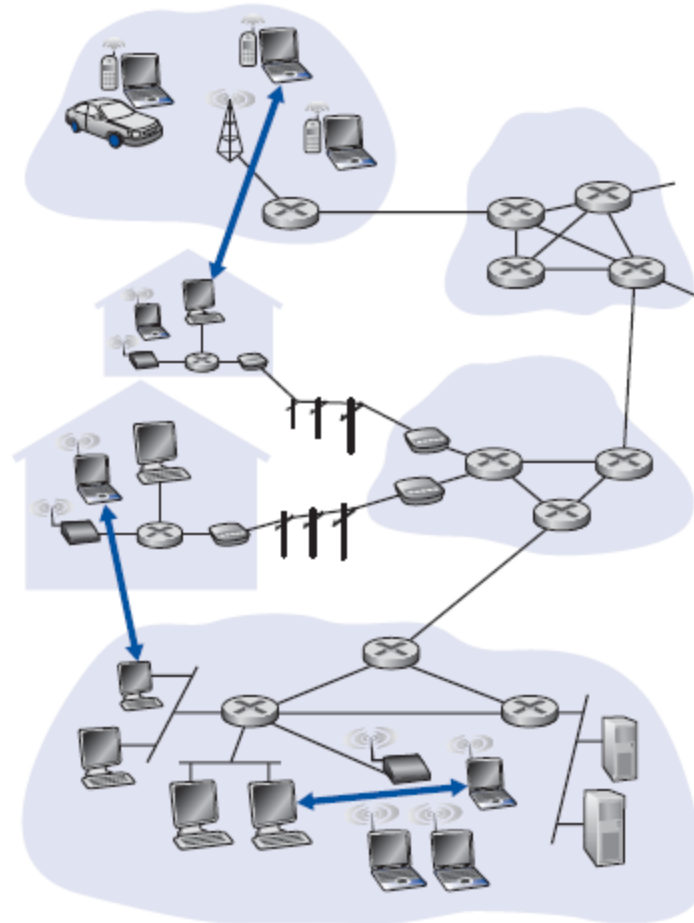
Clients:

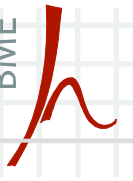
- Kommunizieren mit Servern
- Sporadisch angeschlossen
- Können dynamische IP-Adressen haben
- Kommunizieren nicht direkt miteinander

Reine P2P-Architektur

- *Keine Server*
- Beliebige Endsysteme kommunizieren direkt miteinander
- Peers sind nur sporadisch angeschlossen und wechseln ihre IP-Adresse
- Beispiel: Gnutella, Bitcoin

Gut skalierbar, aber schwer zu warten und zu kontrollieren!





Kombination von Client-Server und P2P

Skype

- P2P-Anwendung für Voice-over-IP
- Zentraler Server: Adresse des Kommunikationspartners finden
- Verbindung zwischen den Klienten: direkt (nicht über einen Server)

Instant Messaging

- Chat zwischen zwei Benutzern: P2P
- Zentralisierte Dienste: Erkennen von Anwesenheit, Zustand, Aufenthaltsort eines Anwenders
 - Benutzer registriert seine IP-Adresse beim Server, sobald er sich mit dem Netz verbindet
 - Benutzer fragt beim Server nach Informationen über seine Freunde und Bekannten

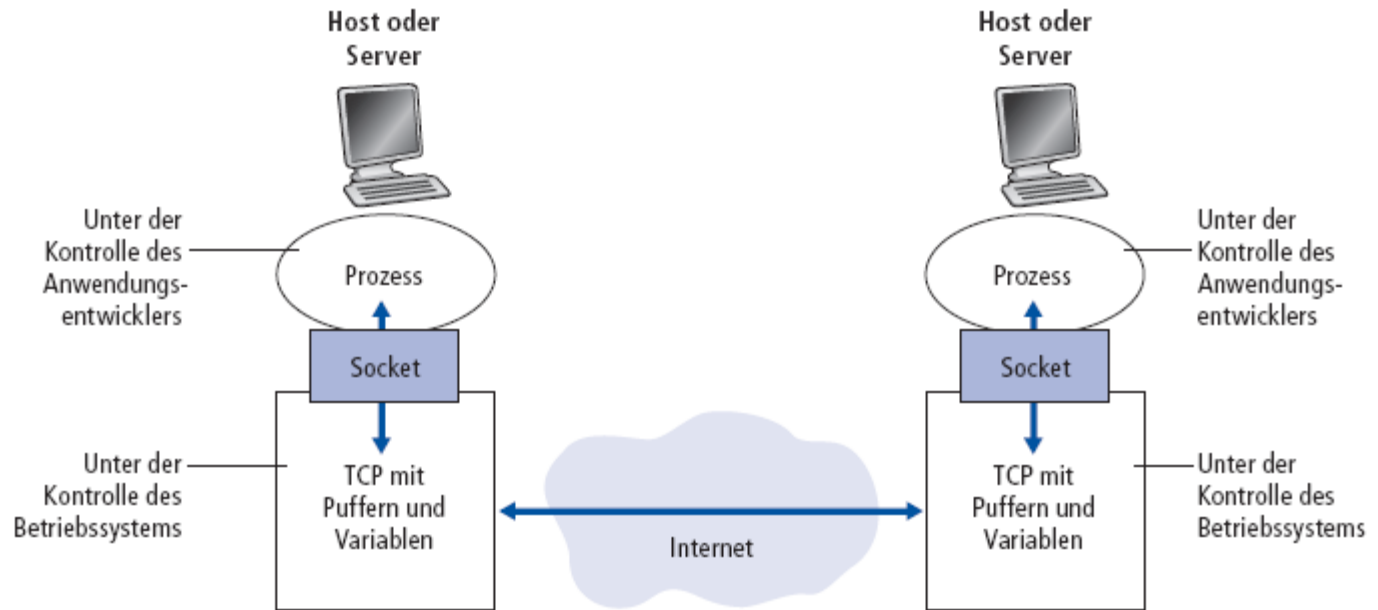
Prozess: Programm, welches auf einem Host läuft

- Innerhalb eines Hosts können zwei Prozesse mit Inter-Prozess-Kommunikation Daten austauschen (durch das Betriebssystem unterstützt)
- Prozesse auf verschiedenen Hosts kommunizieren, indem sie Nachrichten über ein Netzwerk austauschen

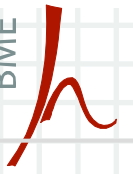
Client-Prozess: Prozess, der die Kommunikation beginnt

Server-Prozess: Prozess, der darauf wartet, kontaktiert zu werden

- Anmerkung: Anwendungen mit einer P2P-Architektur haben Client- und Server-Prozesse



- Prozesse senden/empfangen Nachrichten über einen **Socket**
- Ein Socket lässt sich mit einer (Tür) Port vergleichen
 - Der sendende Prozess schiebt die Nachrichten durch den Port
 - Der sendende Prozess verlässt sich auf die Transportinfrastruktur auf der anderen Seite dem Port, um die Nachricht zum Socket des empfangenden Prozesses zu bringen
- API: (1) Wahl des Transportprotokolls; (2) Einstellen einiger Parameter (mehr dazu später)



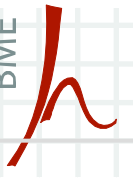
Adressierung von Prozessen

- Um eine Nachricht empfangen zu können, muss ein Prozess identifiziert werden können
- Ein Host besitzt eine eindeutige, 32 Bit lange IP-Adresse
- Frage: Reicht die IP-Adresse, um einen Prozess auf diesem Host zu identifizieren?

Adressierung von Prozessen

- Um eine Nachricht empfangen zu können, muss ein Prozess identifiziert werden können
- Ein Host besitzt eine eindeutige, 32 Bit lange IP-Adresse
- Frage: Reicht die IP-Adresse, um einen Prozess auf diesem Host zu identifizieren?
 - Nein, denn viele Prozesse können auf demselben Host laufen!
- Prozesse werden durch eine **IP-Adresse** UND eine **Portnummer** identifiziert
- Beispiel-Portnummern:
 - HTTP-Server: 80
 - E-Mail-Server: 25
- Um an den Webserver frogstar.hit.bme.hu eine HTTP-Nachricht zu schicken:
 - **IP-Adresse:** 152.66.248.44
 - **Portnummer:** 80

- Well-known ports: 0 – 1023
 - 20 & 21: [File Transfer Protocol](#) (FTP)
 - 22: [Secure Shell](#) (SSH)
 - 23: [Telnet](#) remote login service
 - 25: [Simple Mail Transfer Protocol](#) (SMTP)
 - 53: [Domain Name System](#) (DNS) service
 - 80: [Hypertext Transfer Protocol](#) (HTTP) used in the [World Wide Web](#)
 - 110: [Post Office Protocol](#) (POP3)
 - 119: [Network News Transfer Protocol](#) (NNTP)
 - 143: [Internet Message Access Protocol](#) (IMAP)
 - 161: [Simple Network Management Protocol](#) (SNMP)
 - 443: [HTTP Secure](#) (HTTPS)
- Registered ports: 1024 – 49151
- Unregistered ports: 49152 – 65535



Anwendungsprotokolle bestimmen ...

- Arten von Nachrichten
 - z.B. Request, Response
- Syntax der Nachrichten
 - Welche Felder sind vorhanden und wie werden diese voneinander getrennt?
- Semantik der Nachrichten
 - Bedeutung der Informationen in den Feldern
- Regeln für das Senden von und Antworten auf Nachrichten

Öffentlich verfügbare Protokolle:

- Definiert in RFCs
- Ermöglichen Interoperabilität
- z.B. HTTP, SMTP

Proprietäre Protokolle:

- z.B. Skype

Datenverlust

- Einige Anwendungen können Datenverlust tolerieren (z.B. Audioübertragungen)
- Andere Anwendungen benötigen einen absolut zuverlässigen Datentransfer (z.B. Dateitransfer)

Zeitanforderungen

- Einige Anwendungen (z.B. Internettelefonie oder Netzwerkspiele) tolerieren nur eine sehr geringe Verzögerung

Bandbreite

- Einige Anwendungen (z.B. Multimedia-Streaming) brauchen eine Mindestbandbreite, um zu funktionieren
- Andere Anwendungen verwenden einfach die verfügbare Bandbreite (bandbreitenelastische Anwendungen)

Beispiele für Anforderungen von Anwendungen

Anwendung	Datenverlust	Bandbreite	Echtzeit
Dateitransfer	Kein Verlust	Elastisch	Nein
E-Mail	Kein Verlust	Elastisch	Nein
Web	Kein Verlust	Elastisch (wenige Kbps)	Nein
Internettelefonie/ Bildkonferenz	Toleriert Verluste	Audio: wenige Kbps bis 1 Mbps Video: 10 Kbps bis 5 Mbps	Ja: einige Hundert ms
Gespeichertes Audio/Video	Toleriert Verluste	Wie oben	Ja: wenige Sekunden
Interaktive Spiele	Toleriert Verluste	Wenige Kbps bis 10 Kbps	Ja: einige Hundert ms
Instant Messaging	Kein Verlust	Elastisch	Ja und nein

Dienste der Transportprotokolle

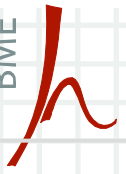
TCP-Dienste:

- *Verbindungsorientierung:* Herstellen einer Verbindung zwischen Client und Server
- *Zuverlässiger Transport* zwischen sendendem und empfangendem Prozess
- *Flusskontrolle:* Sender überlastet den Empfänger nicht
- *Überlastkontrolle:* Bremsen des Senders, wenn das Netzwerk überlastet ist
- *Keine:* Zeit- und Bandbreitengarantien

UDP-Dienste:

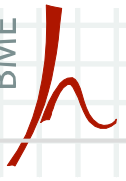
- Unzuverlässiger Transport von Daten zwischen Sender und Empfänger
- *Keine:* Verbindungsorientierung, Zuverlässigkeit, Flusskontrolle, Überlastkontrolle, Zeit- oder Bandbreitengarantien

Frage: Wozu soll das gut sein?
Warum gibt es UDP?



Beispiele aus dem Internet

Anwendung	Anwendungsschichtprotokoll	Zugrunde liegendes Transportprotokoll
E-Mail-Dienst	SMTP [RFC 2821]	TCP
Remote-Terminalzugang	Telnet [RFC 854]	TCP
World Wide Web	HTTP [RFC 2616]	TCP
Dateitransfer	FTP [RFC 959]	TCP
Multimedia-Streaming	HTTP (z.B. YouTube), RTP	TCP oder UDP
Internettelefonie	SIP, RTP oder proprietär (z.B. Skype)	Normalerweise UDP



Kapitel 2: Anwendungsschicht

- 2.1 Grundlagen
- 2.2 Web und HTTP, HTTP/2
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS

Einige Definitionen

- Eine **Webseite** besteht aus **Objekten**
- Objekte können sein: HTML-Dateien, JPEG-Bilder, Java-Applets, Audiodateien, ...
- Eine Webseite hat eine **Basis-HTML-Datei**, die mehrere referenzierte Objekte beinhalten kann
- Jedes Objekt kann durch eine **URL** (Uniform Resource Locator) adressiert werden
- Beispiel für eine URL:

`www.someschool.edu/someDept/pic.gif`

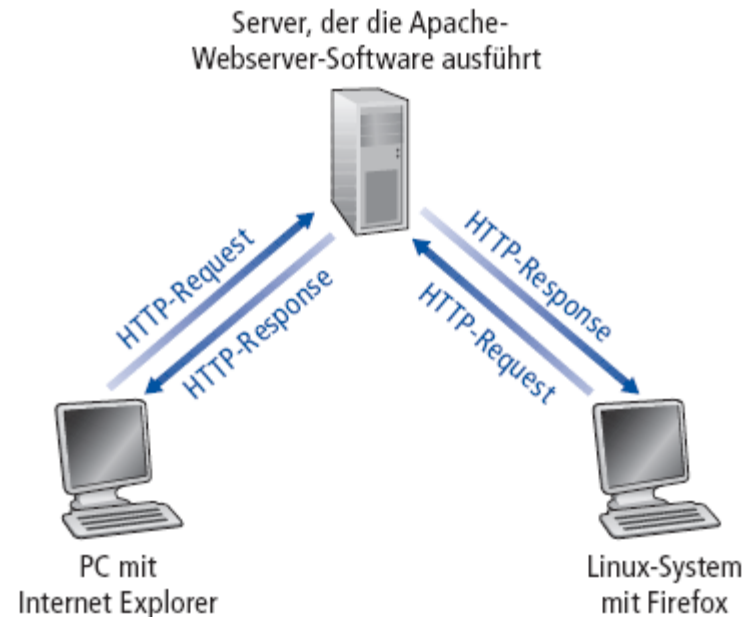
Hostname

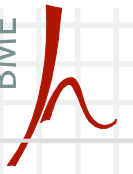
Pfad

PROG: Schreibt ein Program die die Objekte auf einer Webseite listet. Von welchen anderen Seiten kommen die Objekte?

HTTP: HyperText Transfer Protocol

- Anwendungsprotokoll des Web
- Client/Server-Modell
 - **Client:** Browser, der Objekte anfragt, erhält und anzeigt
 - **Server:** Webserver verschickt Objekte auf Anfrage
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068





HTTP: Überblick (Fortsetzung)

Verwendet TCP:

- Client baut mit der Socket-API eine TCP-Verbindung zum Server auf
- Server wartet auf **Port 80**
- Server nimmt die TCP-Verbindung des Clients an
- HTTP-Nachrichten (Protokollnachrichten der Anwendungsschicht) werden zwischen Browser (HTTP-Client) und Webserver (HTTP-Server) ausgetauscht
- Die TCP-Verbindung wird geschlossen

HTTP ist “zustandslos”

- Server merkt sich keine Informationen über frühere Anfragen von Clients

Protokolle, die einen Zustand verwalten, sind komplex!

- Der Zustand muss gespeichert und verwaltet werden
- Wenn Server oder Client abstürzen, dann muss der Zustand wieder synchronisiert werden

Nichtpersistentes HTTP

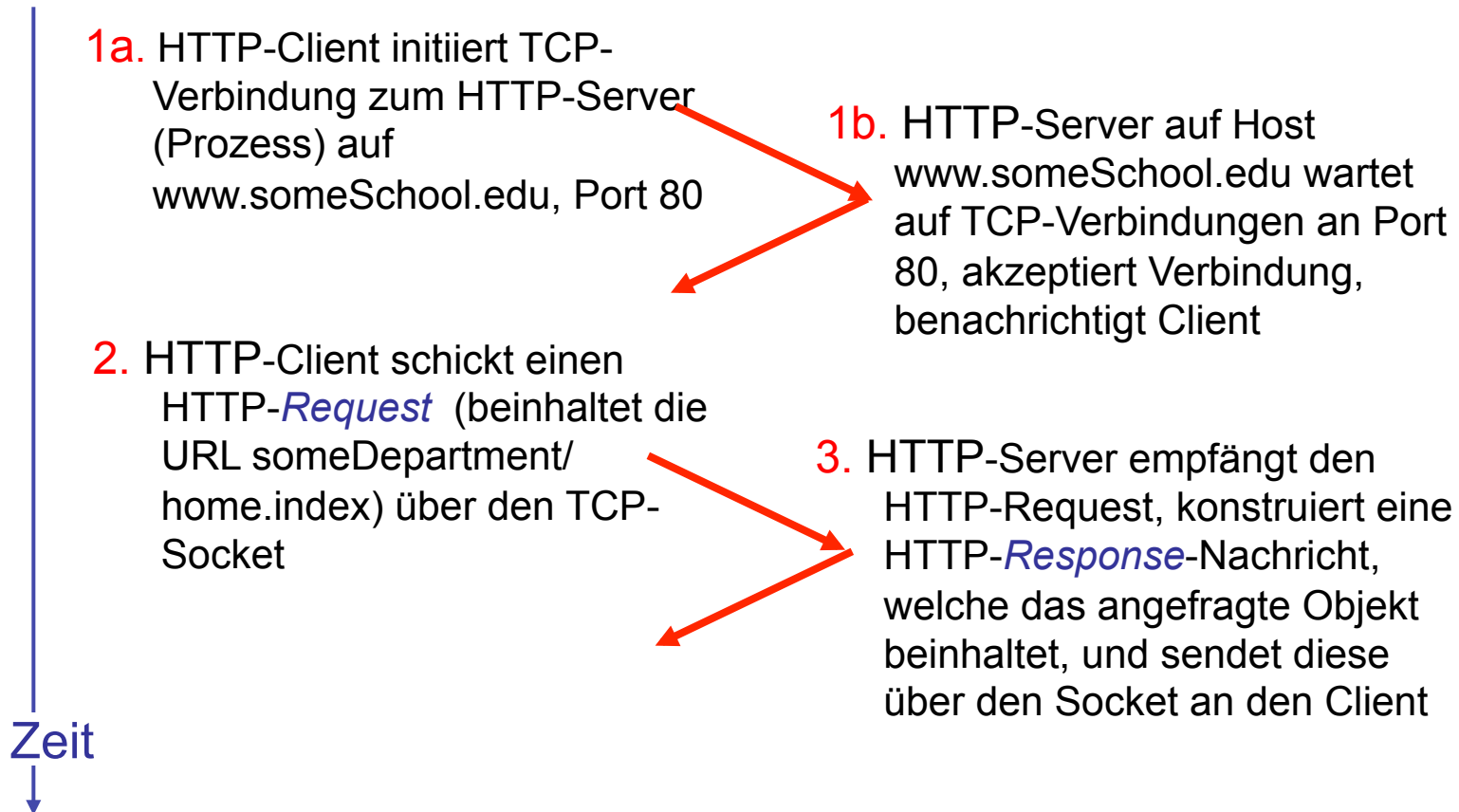
- Maximal ein Objekt wird über eine TCP-Verbindung übertragen
- HTTP/1.0 verwendet nichtpersistentes HTTP

Persistentes HTTP

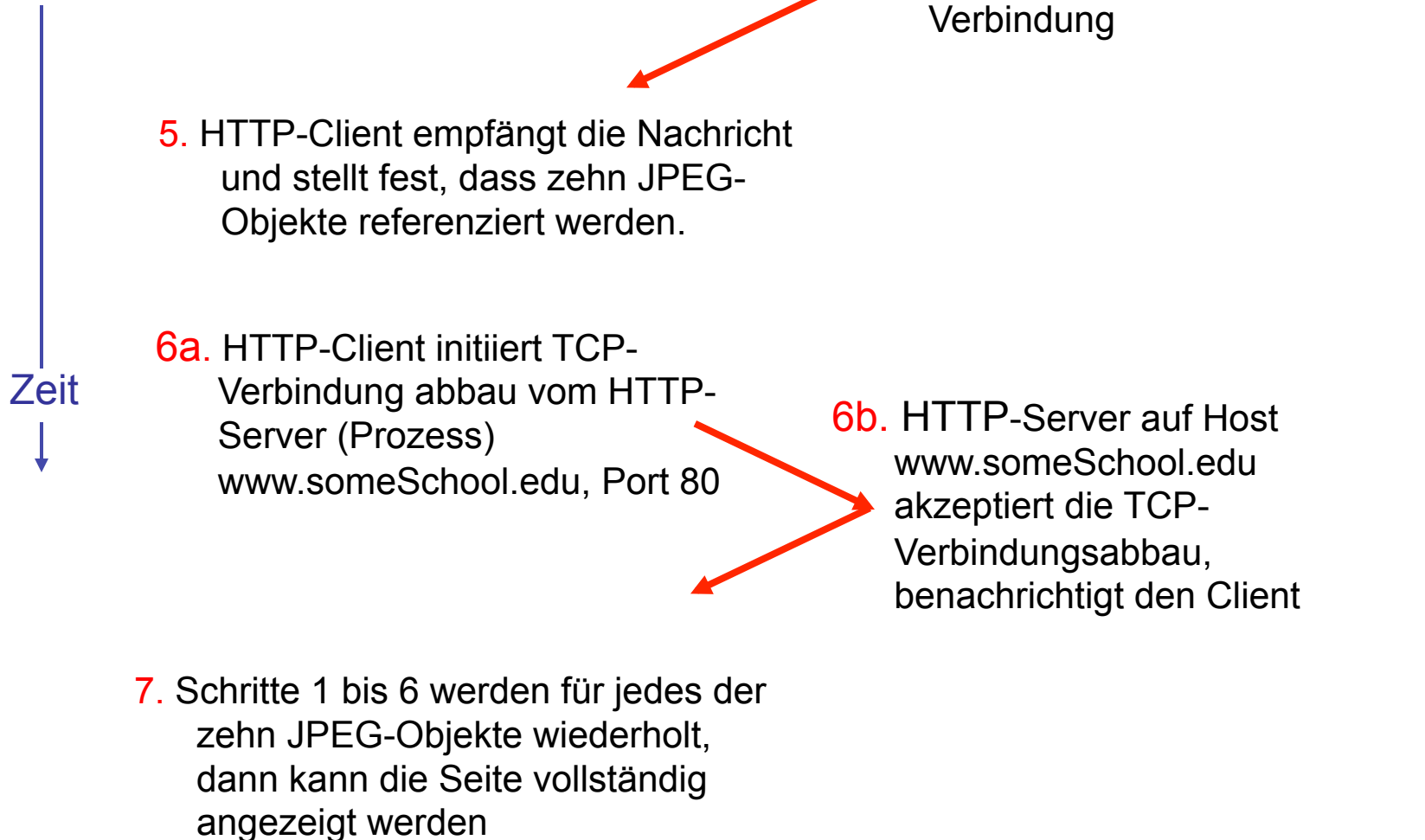
- Mehrere Objekte können über eine TCP-Verbindung übertragen werden
- HTTP/1.1 verwendet standardmäßig persistentes HTTP
- keep-alive in 1999 eingeführt

Nichtpersistentes HTTP

Es soll folgende URL geladen werden: `www.someSchool.edu/
someDepartment/home.index`



Nichtpersistentes HTTP (Forts.)



Nichtpersistentes HTTP: Verzögerung

Definition von RTT (Round Trip Time):

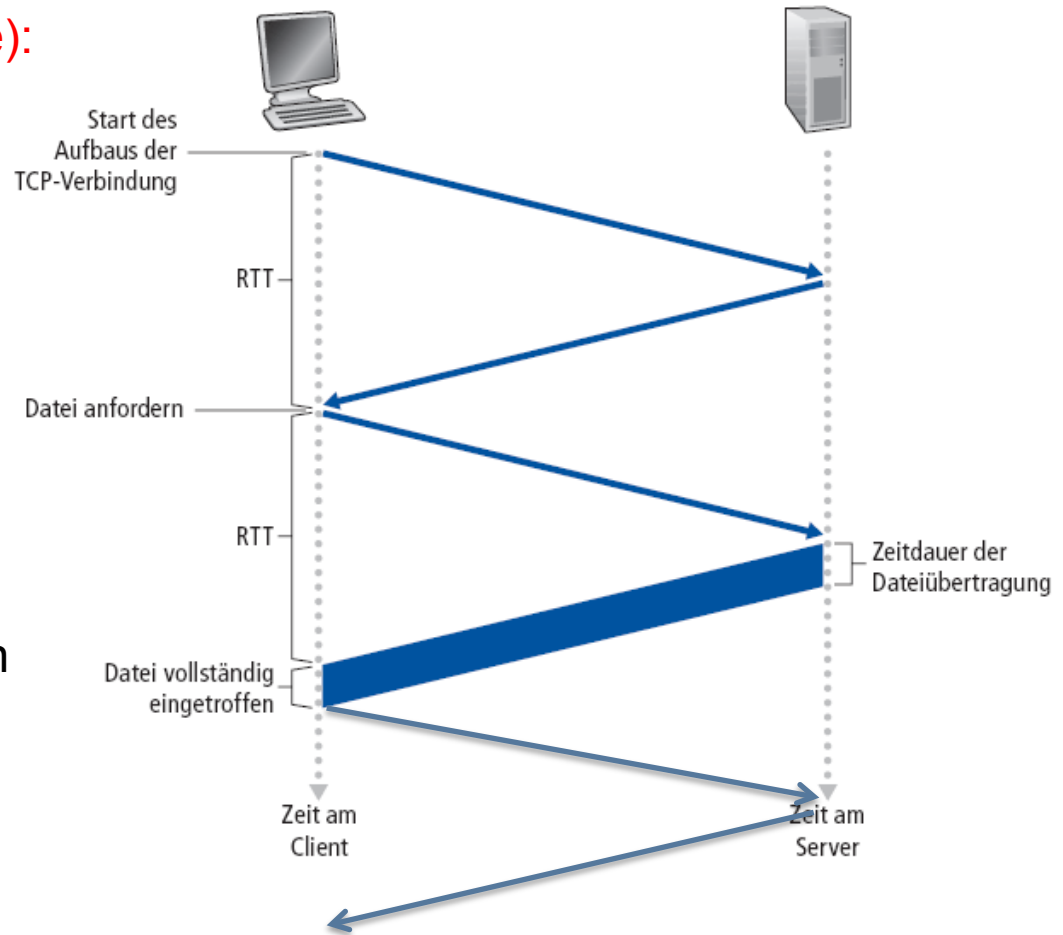
Zeit, um ein kleines Paket vom Client zum Server und zurück zu schicken

Verzögerung:

- Eine RTT für den TCP-Verbindungsaufbau
- Eine RTT für den HTTP-Request, bis das erste Byte der HTTP-Response beim Client ist
- Zeit für das Übertragen der Daten auf der Leitung
- Zeit für Verbindungsabbau

Zusammen = $Nr_Objekt * (3 \text{ RTT} + \text{Übertragungsverzögerung})$

TCP-Verbindung für jedes Objekt nochmal aufgebaut!



Probleme mit nichtpersistentem HTTP:

- 2 RTTs plus pro Objekt
- Aufwand im Betriebssystem für jede TCP-Verbindung
- Browser öffnen oft mehrere parallele TCP-Verbindungen, um die referenzierten Objekte zu laden

Persistentes HTTP

- Server lässt die Verbindung nach dem Senden der Antwort offen
- Nachfolgende HTTP-Nachrichten können über dieselbe Verbindung übertragen werden

Persistent *ohne* Pipelining:

- Client schickt neuen Request erst, nachdem die Antwort auf den vorangegangenen Request empfangen wurde
- Eine RTT für jedes referenzierte Objekt

Persistent mit Pipelining:

- Standard in HTTP/1.1
- Client schickt Requests, sobald er die Referenz zu einem Objekt findet
- Idealerweise wird nur wenig mehr als eine RTT für das Laden aller referenzierten Objekte benötigt

Persistentes HTTP ohne Pipelining : Verzögerung

Definition von RTT (Round Trip Time):

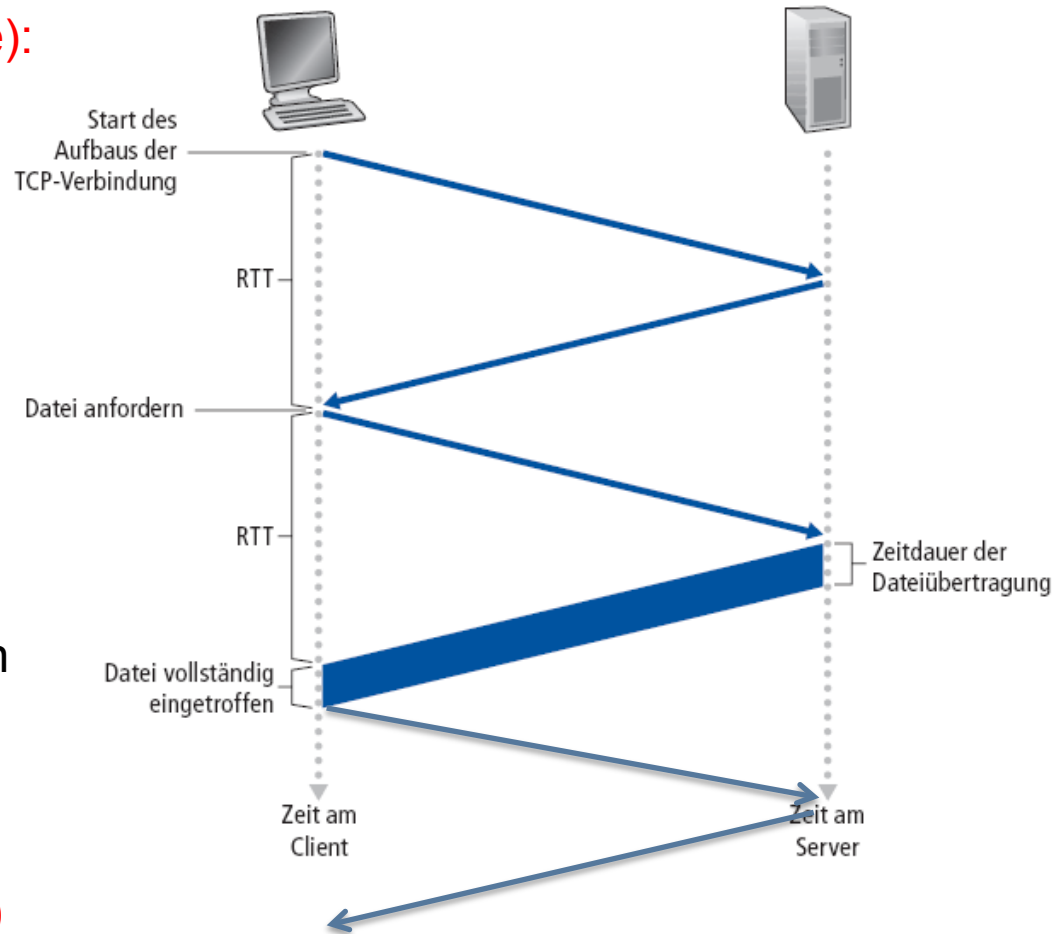
Zeit, um ein kleines Paket vom Client zum Server und zurück zu schicken

Verzögerung:

- Eine RTT für den TCP-Verbindungsaufbau
- Eine RTT für den HTTP-Request, bis das erste Byte der HTTP-Response beim Client ist
- Zeit für das Übertragen der Daten auf der Leitung
- Zeit für Verbindungsabbau

Zusammen = $2 \text{ RTT} + \text{Nr_Objekte} * (\text{RTT} + \text{Übertragungsverzögerung})$

TCP Verbindungs aufbau und abbau nur einmal ausgeführt!



Persistentes HTTP mit Pipelining : Verzögerung

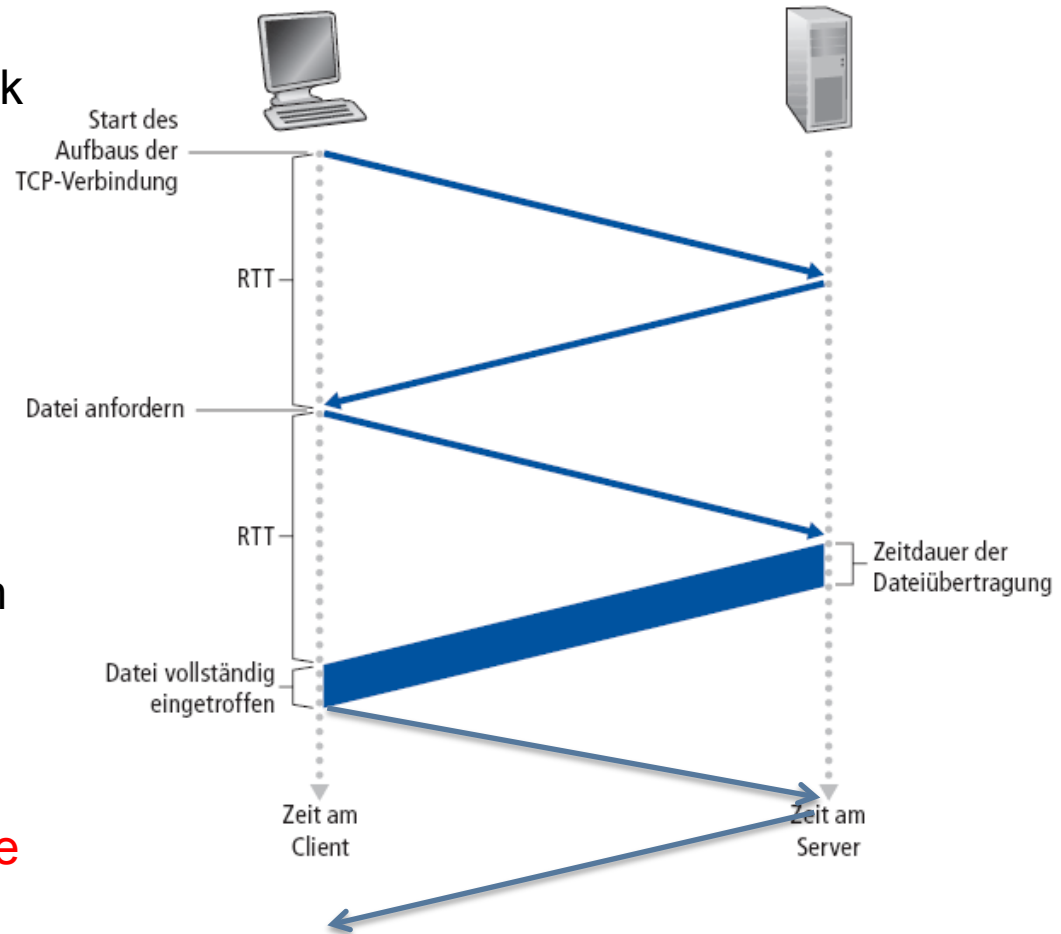
Definition von RTT (Round Trip Time): Zeit, um ein kleines Paket vom Client zum Server und zurück zu schicken

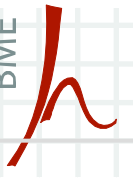
Verzögerung:

- Eine RTT für den TCP-Verbindungsaufbau
- Eine RTT für den HTTP-Request, bis das erste Byte der HTTP-Response beim Client ist
- Zeit für das Übertragen der Daten auf der Leitung
- Zeit für Verbindungsabbau

Zusammen = 2 RTT + (RTT + Übertragung) + RTT + Nr_Objekte * Übertragung

TCP Verbindungs aufbau und abbau nur einmal ausgeführt!



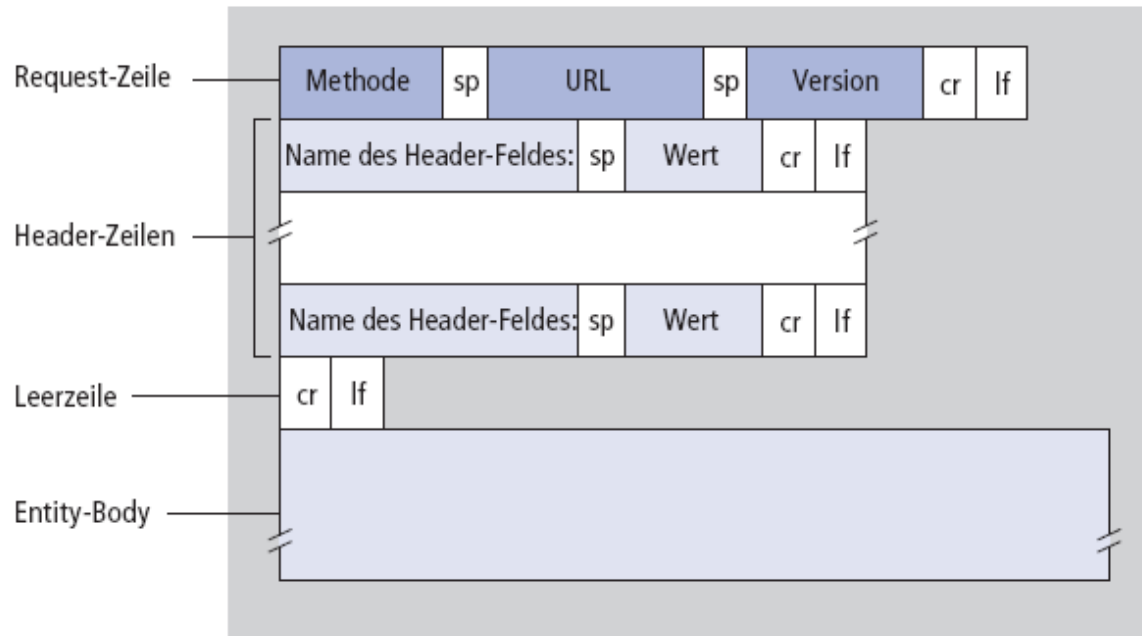


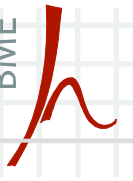
Pipelining nicht benutzt

- Head-of-Line blocking (kommt später)

HTTP-Request Nachricht

- Zwei Arten von HTTP-Nachrichten: *Request*, *Response*
- **HTTP-Request-Nachricht:**
 - ASCII (vom Menschen leicht zu lesen)





HTTP-Request Nachricht

Request-Zeile
(GET, POST,
HEAD commands)

Header-Zeilen

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
```

Zusätzlicher Wagenrücklauf +
Zeilenvorschub zeigt das Ende der Nachricht an

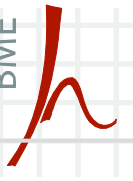
Post-Anweisung:

- Webseiten beinhalten häufig Formulare, in denen Eingaben erfolgen sollen
- Eingaben werden zum Server im Datenteil (entity body) der Post-Anweisung übertragen

Get-Anweisung:

- Eingabe wird als Bestandteil der URL übertragen:

`www.somesite.com/animalsearch?monkeys&banana`



Typen von Anweisungen

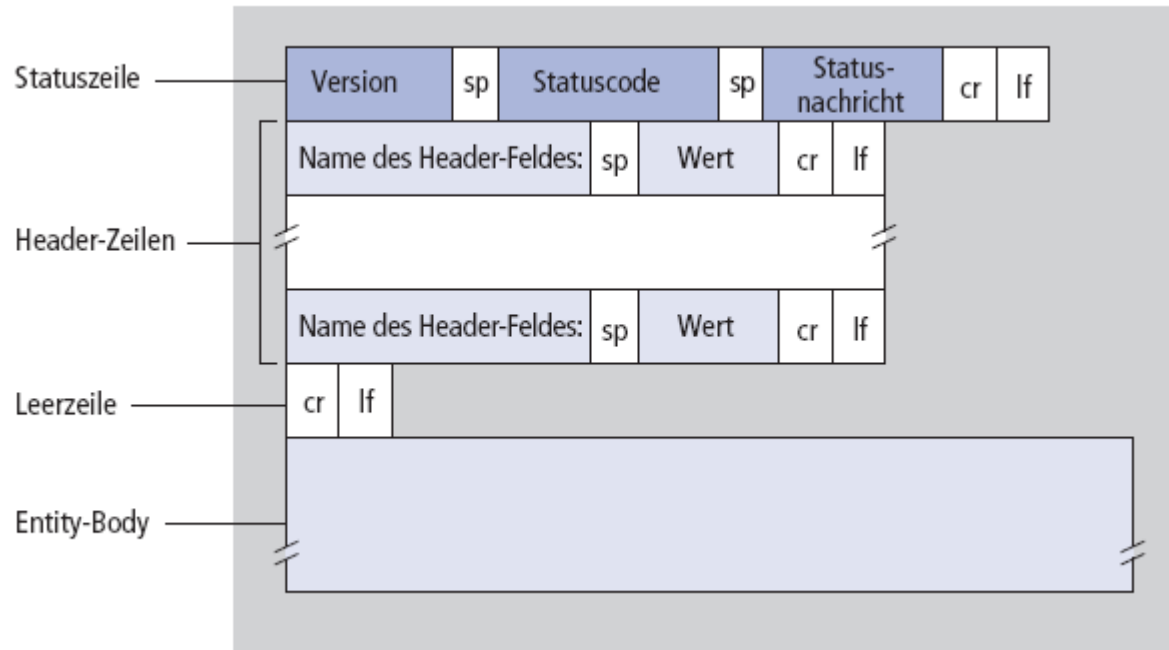
HTTP/1.0

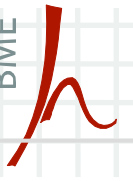
- GET
- POST
- HEAD
 - Bittet den Server, nur die Kopfzeilen der Antwort (und nicht das Objekt) zu übertragen

HTTP/1.1

- GET, POST, HEAD
- PUT
 - Lädt die im Datenteil enthaltene Datei an die durch eine URL bezeichnete Position hoch
- DELETE
 - Löscht die durch eine URL angegebene Datei auf dem Server

HTTP-Response-Nachricht: Format





HTTP-Response-Nachricht

Statuszeile
(Statuscode,
Statusnachricht)

Header-Zeilen

HTTP/1.1 200 OK

Connection close

Date: Thu, 06 Aug 1998 12:00:15 GMT

Server: Apache/1.3.0 (Unix)

Last-Modified: Mon, 22 Jun 1998

Content-Length: 6821

Content-Type: text/html

data data data data data ...

Entity Body: Daten, z.B. die
angefragte HTML-Datei

Statuscodes für HTTP-Response

In der ersten Zeile der Response-Nachricht
Einige Beispiele für Statuscodes:

200 OK

- Request war erfolgreich, gewünschtes Objekt ist in der Antwort enthalten

301 Moved Permanently

- Gewünschtes Objekt wurde verschoben, neue URL ist in der Antwort enthalten

400 Bad Request

- Request-Nachricht wurde vom Server nicht verstanden

403 Forbidden

- Request-Nachricht kann nicht gesendet werden, kein Zugang
- z.B. 403.14 - Directory listing denied

404 Not Found

- Gewünschtes Objekt wurde nicht gefunden

505 HTTP Version Not Supported

PROG: Schreibt ein HTTP Server der den Code 418 (Im a teapot) implementiert



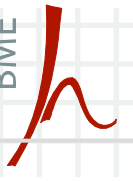
ERROR 404

Page not available But Justin is.

Justin is a Mint developer who likes slow cars, sharp crayons, reheated pizza and awkward silence. Email him at [justin \[at \] mint.com](mailto:justin[at]mint.com).

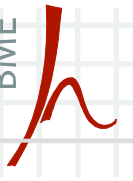
But if you're more interested in personal finance than in Justin, try the links below:





HTTP/2

- die Verzögerung vermindern
- HOL blocking eliminieren
- Multiplexierung möglich



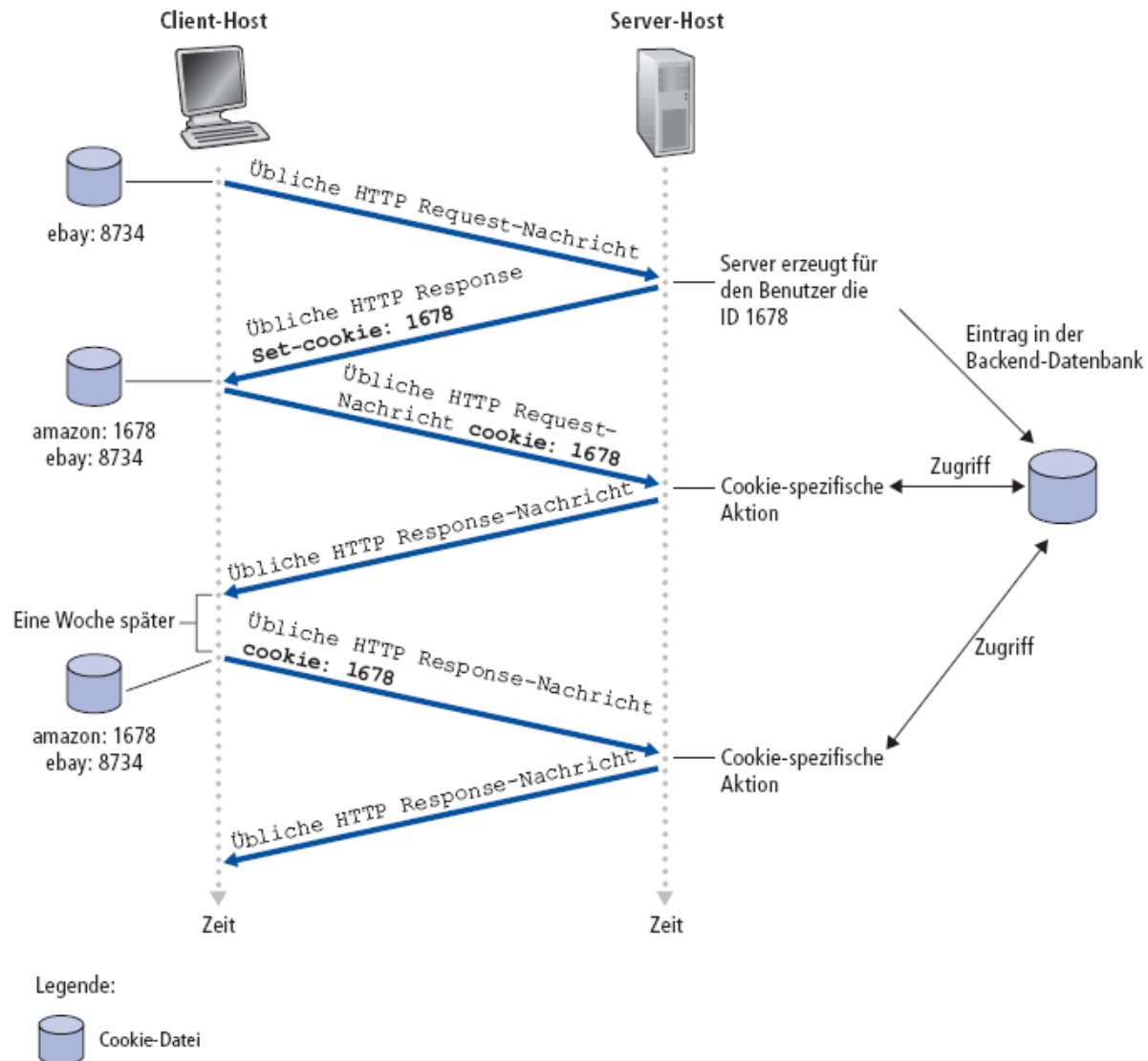
Zustand halten: Cookies

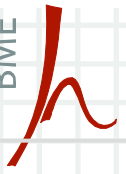
Viele wichtige Websites verwenden Cookies

Vier Bestandteile:

- 1) Cookie-Kopfzeile in der HTTP-Response-Nachricht
- 2) Cookie-Kopfzeile in der HTTP-Request-Nachricht
- 3) Cookie-Datei, die auf dem Rechner des Anwenders angelegt und vom Browser verwaltet wird
- 4) Backend-Datenbank auf dem Webserver

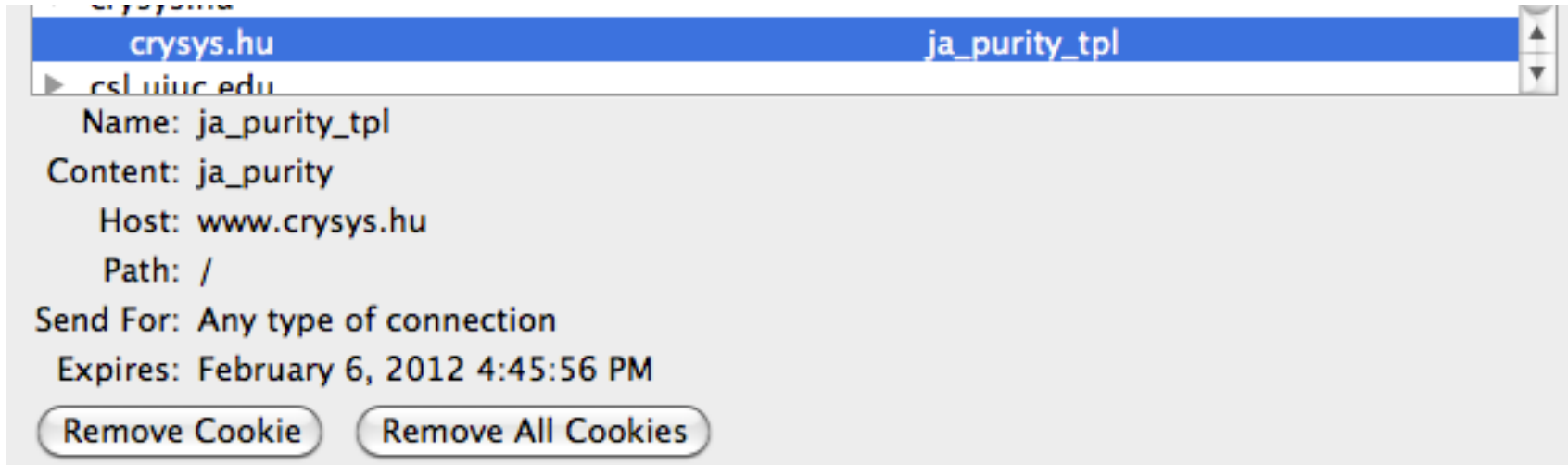
Zustand halten: Cookies



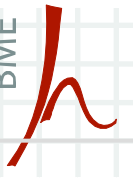


Zustand halten: Cookies

- Set-Cookie: CO=kiskacs;
- Set-Cookie: CO=kiskacs; Expires=Wed, 09 Jun 2021 10:18:14 GMT
- Set-Cookie: CO=kiskacs; Domain=.google.com; Path=/; Expires=Wed, 09 Jun 2021 10:18:14 GMT; httpOnly



- Sieht Eure Browser Cookie-Liste und
- **PROG:** Schreibt ein Programm der Euch warnt wenn eine Seite ein Cookie setzen möchte



Zustand halten: Cookies

Einsatz von Cookies:

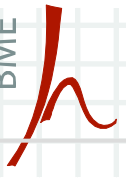
- Autorisierung
- Einkaufswagen
- Empfehlungen
- Sitzungszustand (z.B. bei Web-E-Mail)

Cookies und Privatsphäre:

- Cookies ermöglichen es Websites, viel über den Anwender zu lernen:
 - Formulareingaben (Name, E-Mail-Adresse)
 - Besuchte Seiten

Alternativen, um Zustand zu halten:

- In den Endsystemen: Zustand wird im Protokoll auf dem Client oder Server gespeichert und für mehrere Transaktionen verwendet
- Cookies: HTTP-Nachrichten beinhalten den Zustand



Zustand halten: Super Cookies

- Users sind gewarnt von Cookies (EU Initiative)
- neue Methoden von User-Befolgung
- Super-Cookies – viel gefährlicher als Cookies
 - Adobe Flash Cookies
 - DOM storage in HTML5
 - bis zum 5-10MB
 - nicht so einfach zu löschen

Zustand halten: Browser Fingerprinting

- Browser-fingerprinting

eff.org https://panoptick.eff.org/index.php?action=log&js=yes

comp moving courses house

InfoWorld Home

JANUARY 29, 2010

EFF only browser

A browser's and can pote

By Jeremy Kirk | I

Panoptick

How Unique – and Trackable – Is Your Browser?

Your browser fingerprint **appears to be unique** among the 1,439,950 tested so far.

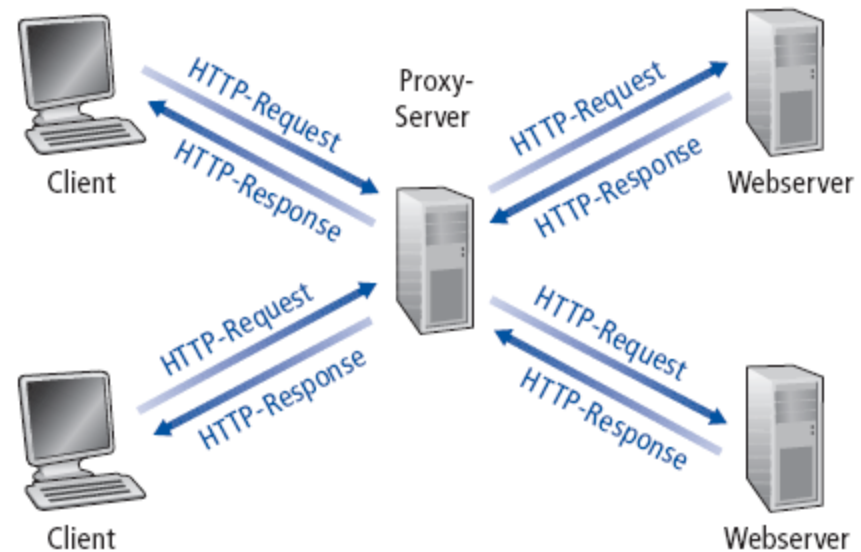
Currently, we estimate that your browser has a fingerprint that conveys **at least 20.46 bits of identifying information**.

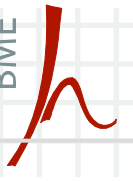
The measurements we used to obtain this result are listed below. You can read more about our methodology, statistical results, and some defenses against fingerprinting in **this article**.

Web-Caches (Proxy-server)

Ziel: Anfragen des Clients ohne den ursprünglichen Webserver beantworten

- Benutzer konfiguriert Browser: Webzugriff über einen Cache
- Browser sendet alle HTTP-Requests an den Cache
 - Objekt im Cache: Cache gibt Objekt zurück
 - Sonst: Cache fragt das Objekt vom ursprünglichen Server an und gibt es dann an den Client zurück





Mehr zum Thema Web-Caching

- Cache arbeitet als Client UND als Server
- Üblicherweise ist der Cache beim ISP installiert (Universität, Firma, ISP für Privathaushalte)

Warum Web-Caching?

- Verringert die Antwortzeit
- Verringert den Datenverkehr auf der Zugangsleitung
- Bei Existenz vieler Caches: „arme“ Inhaltsanbieter können ihre Inhalte gut verbreiten (Ähnliches kann durch P2P-Filesharing erreicht werden)

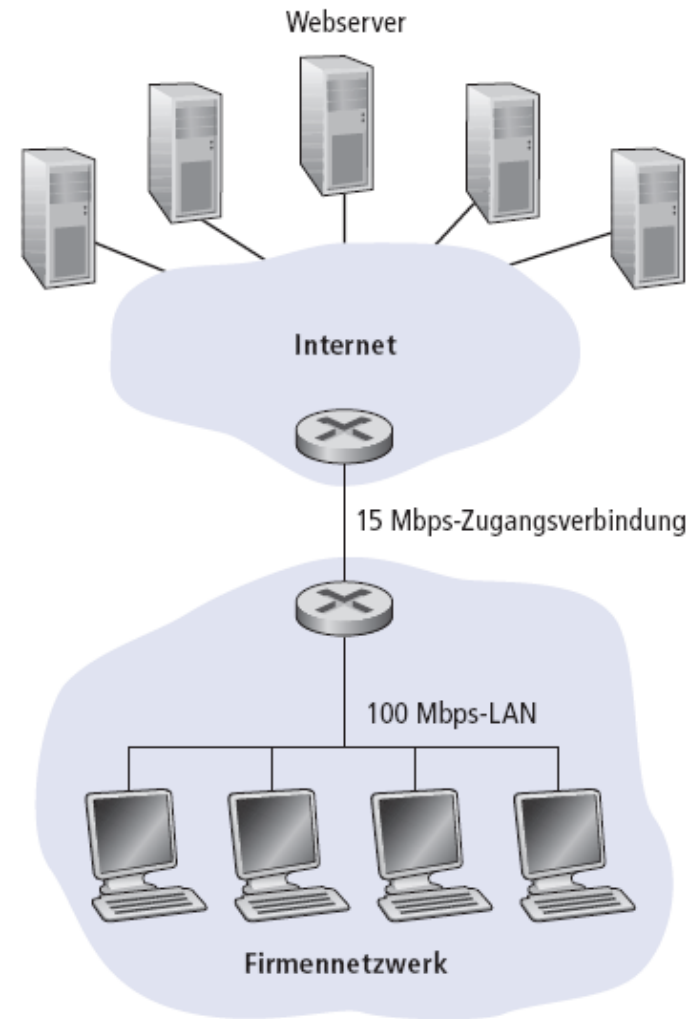
Beispiel für Web-Caching

Annahmen

- Durchschnittliche Größe eines Objektes = 1Mbit
- Durch. Rate von Anfragen aller Webbrowser der Firma = 15/s
- Verzögerung v. Router d. Firma zum Server und zurück = 2 sec

Resultat

- Auslastung des LAN = 15%
- Auslastung der Zugangsleitung = 100%
- Verzögerung = Internet + Zugangsleitung + LAN
= 2s + Minuten + Millisekunden



Beispiel für Web-Caching

Mögliche Lösung

- Erhöhen der Bandbreite der Zugangsleitung auf 100 Mbit/s

Resultat

- Auslastung des LAN = 15%
- Auslastung der Zugangsleitung = 15%
- Verzögerung = Internet + Zugangsleitung + LAN
= 2 sec + Millisekunden + Millisekunden
- Oft sehr teuer!

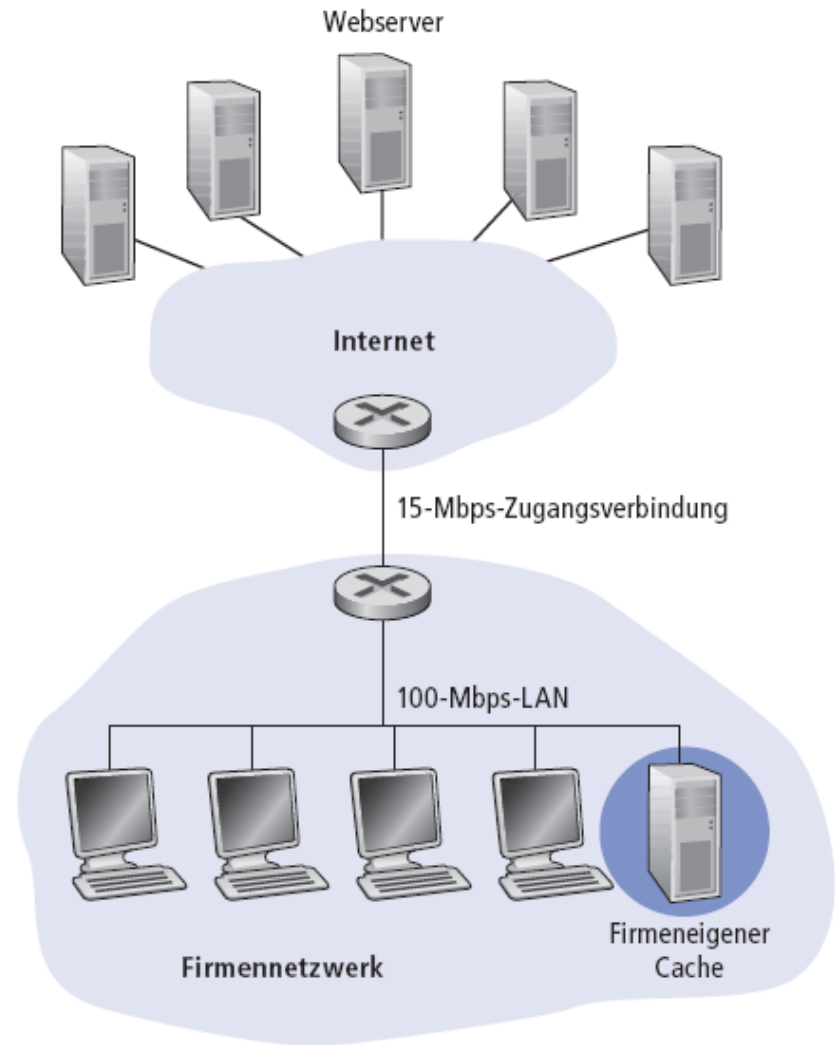
Beispiel für Web-Caching

Mögliche Lösung

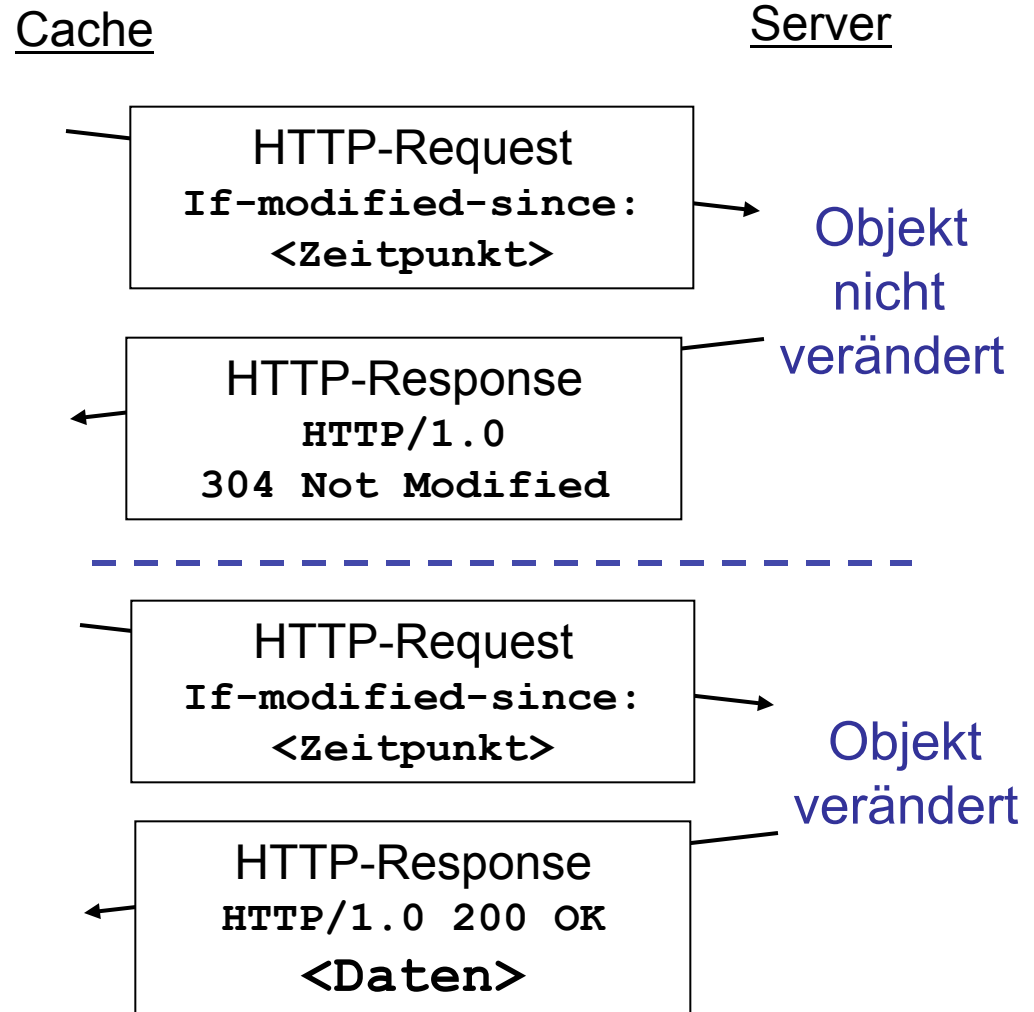
- Web-Cache installieren
- Annahme: Trefferrate = 0,4

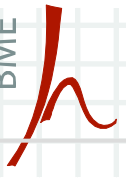
Resultat

- 40% den Anfragen werden nahezu sofort beantwortet
- 60% den Anfragen werden durch normale Webserver beantwortet
- Auslastung der Zugangsleitung auf 60% reduziert, Verzögerungen werden vernachlässigbar klein (z.B. 10 msec)
- $E[\text{Verzögerung}] = \text{Internet} + \text{Zugangsleitung} + \text{LAN} = 0,6 \cdot (2,01\text{s}) + 0,4 \cdot \text{Millisekunden} < 1,4\text{s}$



- **Ziel:** Objekt nicht senden, wenn der Cache eine aktuelle Version besitzt
- Cache: Angeben des Änderungsdatums der gespeicherten Version (kann der HTTP-Response entnommen werden) durch folgende Zeile im HTTP-Request:
`If-modified-since: <date>`
- Server: Response enthält kein Objekt, wenn die Version im Cache aktuell ist:
`HTTP/1.0 304 Not Modified`

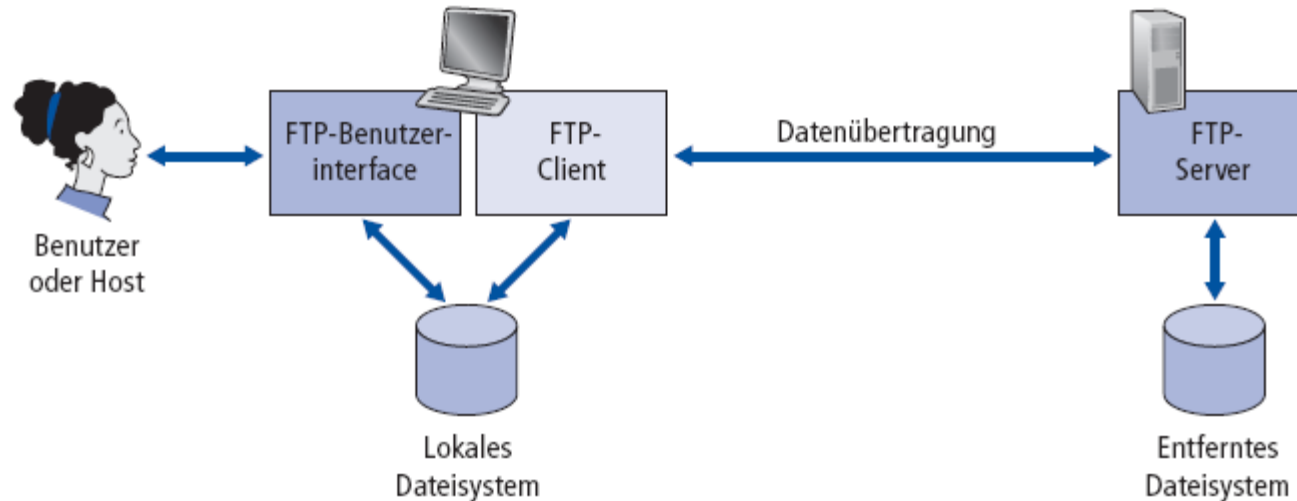




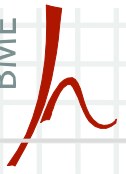
Kapitel 2: Anwendungsschicht

- 2.1 Grundlagen
- 2.2 Web und HTTP, HTTP/2
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS

FTP: Das File Transfer Protocol

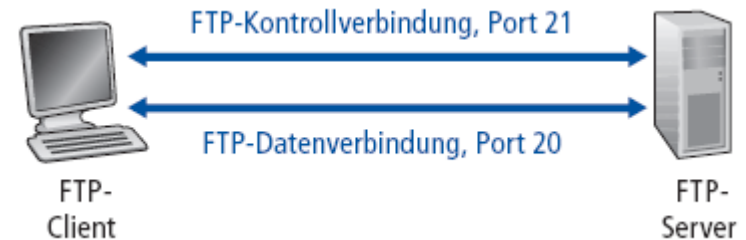


- Übertragen einer Datei von/zum einem entfernten Rechner
- Client/Server-Modell
 - *Client*: Seite, die den Transfer initiiert (vom oder zum entfernten Rechner)
 - *Server*: entfernter Rechner
- FTP: RFC 959
- FTP-Server: TCP Port 21

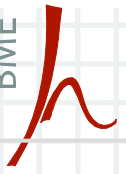


FTP: Verschiedene Kanäle für Kontroll- und Datenverbindungen

- FTP-Client kontaktiert den FTP-Server auf Port 21, wobei er TCP als Transportprotokoll nutzt
- Client autorisiert sich über die Kontrollverbindung
- Client betrachtet das entfernte Verzeichnis, indem er Kommandos über die Kontrollverbindung schickt
- Empfängt der Server ein Kommando für eine Dateiübertragung, öffnet der Server eine TCP-Datenverbindung zum Client
- Nach der Übertragung einer Datei schließt der Server die Verbindung



- Server öffnet eine zweite TCP-Datenverbindung, um noch eine Datei zu übertragen
- Kontrollverbindung: **“out of band”**
- FTP Server hält „Statusinformationen“ vor: aktuelles Verzeichnis, frühere Authentifizierung



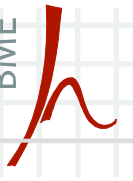
FTP-Kommandos, Antworten

Kommandos:

- Als ASCII-Text über die Kontrollverbindung
- **USER *username***
- **PASS *password***
- **LIST** gibt eine Liste der Dateien im aktuellen Verzeichnis zurück
- **RETR *filename*** lädt eine entfernte Datei auf den lokalen Rechner
- **STOR *filename*** überträgt eine lokale Datei auf den entfernten Rechner

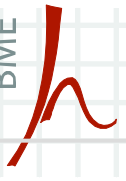
Antworten:

- Statuscode und Erläuterung (wie bei HTTP)
- 331 Username OK, password required
- 125 data connection already open; transfer starting
- 425 Can't open data connection
- 452 Error writing file



Andere Anwendungen

- SCP – port 22
- SFTP = SSH + FTP – port 22
- rsync – port 873



Kapitel 2: Anwendungsschicht

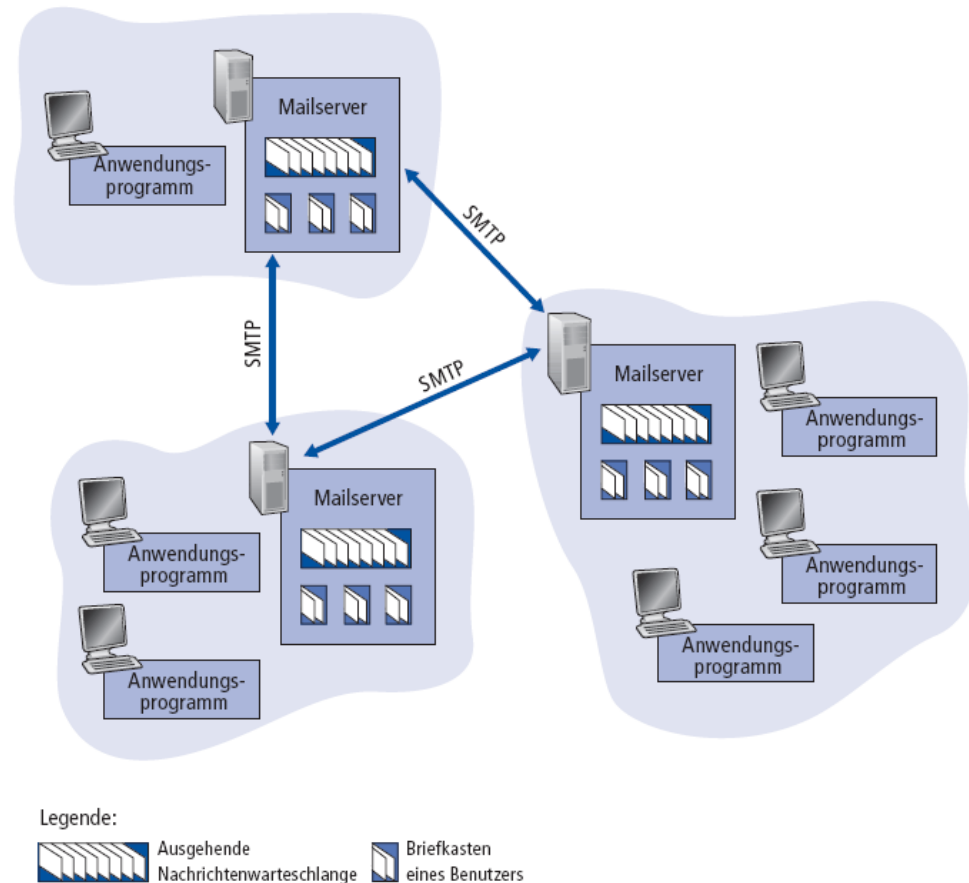
- 2.1 Grundlagen
- 2.2 Web und HTTP, HTTP/2
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS

Drei Hauptbestandteile:

- Anwendungsprogramm
- Mailserver
- Übertragungsprotokoll: SMTP (Simple Mail Transfer Protocol)

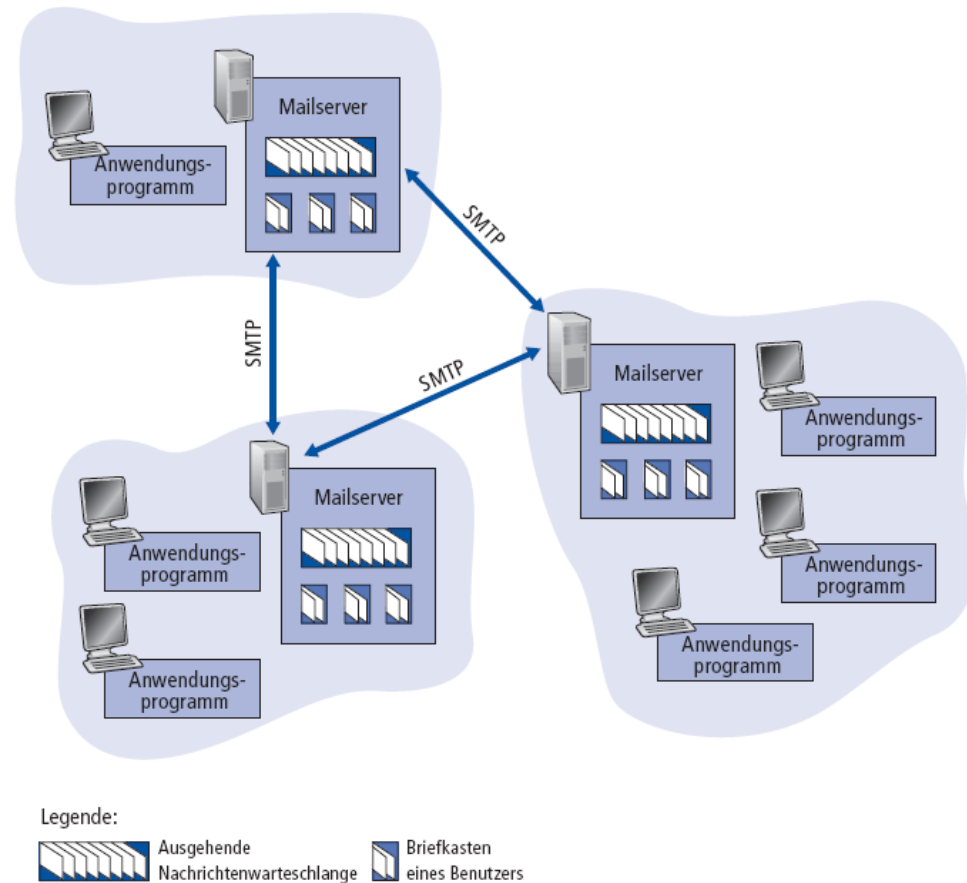
Anwendungsprogramm:

- Auch "Mail Reader"
- Erstellen, Editieren, Lesen von E-Mail-Nachrichten
- z.B. Outlook, Thunderbird, Apple Mail, pine
- Eingehende und ausgehende Nachrichten werden auf dem Server gespeichert



Mailserver

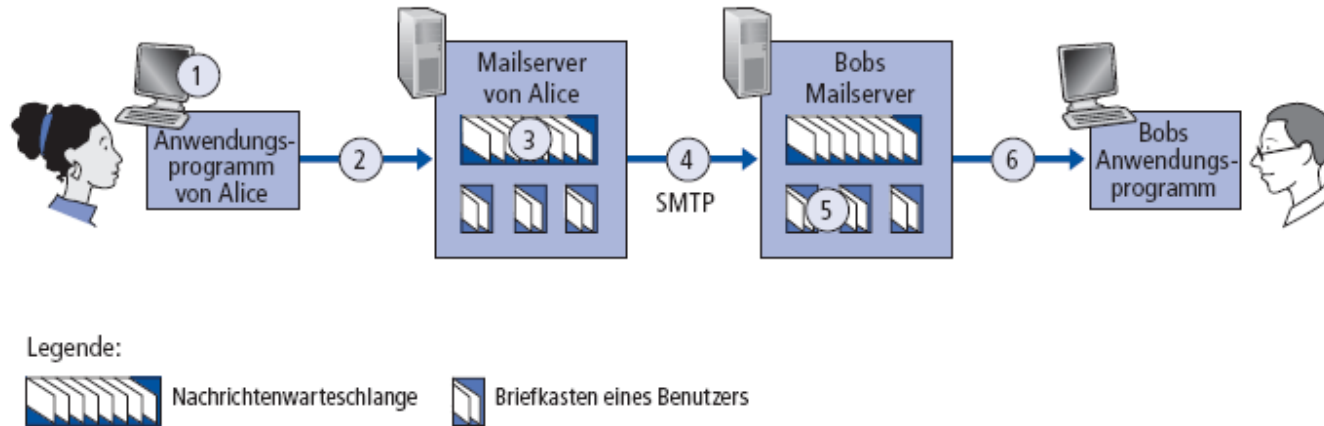
- Die **Mailbox** enthält die eingehenden Nachrichten eines Benutzers
- Die **Warteschlange** für ausgehende Nachrichten enthält die noch zu sendenden E-Mail-Nachrichten
- **SMTP** wird verwendet, um Nachrichten zwischen Mailservern auszutauschen
 - Client: sendender Mailserver
 - Server: empfangender Mailserver



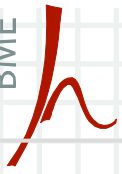
Electronic Mail: SMTP [RFC 2821]

- Ursprüngliche Version aus dem Jahr 1982
- TCP wird zum zuverlässigen Transport von E-Mail-Nachrichten vom Client zum Server (Port 25) verwendet
- Direkter Transport der Nachrichten: vom sendenden Server zum empfangenden Server
- Drei Phasen des Mail-Versands: analog zu einer Unterhaltung
 - Handshaking (Begrüßung)
 - Transfer of Messages (Austausch von Informationen)
 - Closure (Verabschiedung)
- Interaktion basiert auf dem Austausch von Befehlen (Commands) und Antworten (Responses)
 - **Command:** ASCII-Text
 - **Response:** Statuscode und Bezeichnung
- Nachrichten müssen in 7-Bit-ASCII kodiert sein

Beispiel: Ein Mail von Alice zu Bob

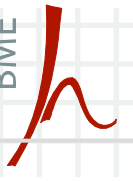


- 1) Alice verwendet ihr Anwendungsprogramm (**MUA Alice, Mail User Agent**), um eine Nachricht an `bob@someschool.edu` zu erstellen
- 2) Alices Anwendung versendet die Nachricht an ihren Mail-Server (**MSA, Message Sender Agent**); Nachricht wird in der Warteschlange gespeichert (**MTA Alice, Message Transfer Agent**)
- 3) Alices Mailserver öffnet als Client eine TCP-Verbindung zu Bobs Mailserver
- 4) SMTP-Client versendet die Nachricht von Alice über die TCP-Verbindung
- 5) Bobs Mailserver empfängt die Nachricht (**MTA Bob**) von Alices Mailserver und speichert diese in Bobs Mailbox (**MDA, Message Delivery Agent**)
- 6) Bob verwendet (irgendwann) sein Anwendungsprogramm (**MUA Bob**) und liest die Nachricht



Beispiel für eine SMTP-Sitzung

```
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
```

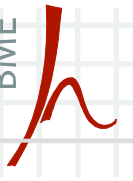


SMTP selbst ausprobieren:

- `telnet servername 25`
- Der Server sollte mit dem Code 220 antworten
- Eingeben der Befehle HELO, MAIL FROM, RCPT TO, DATA, QUIT

So kann man eine E-Mail ohne Verwendung eines Anwendungsprogramms versenden

Irgendwelche Probleme?



SMTP verbessert

- ESMTP – sicheres Senden mit Port 587
- Klient Authentikation
- open relay
 - ein Server, der Alles weitersendet
 - sehr gefährlich, normalerweise auf die Schwartzliste
- einige ISPs blocken port 25 – Benutzer müssen die SMTP Server von den ISP benutzen

Probleme?

SMTP: Zusammenfassung

- SMTP verwendet eine dauerhafte Verbindung für den Versand von E-Mails
- SMTP verwendet sowohl für Header als auch für Daten 7-Bit-ASCII
- Ein SMTP-Server verwendet `CRLF.CRLF`, um das Ende einer Nachricht zu signalisieren

Vergleich mit HTTP:

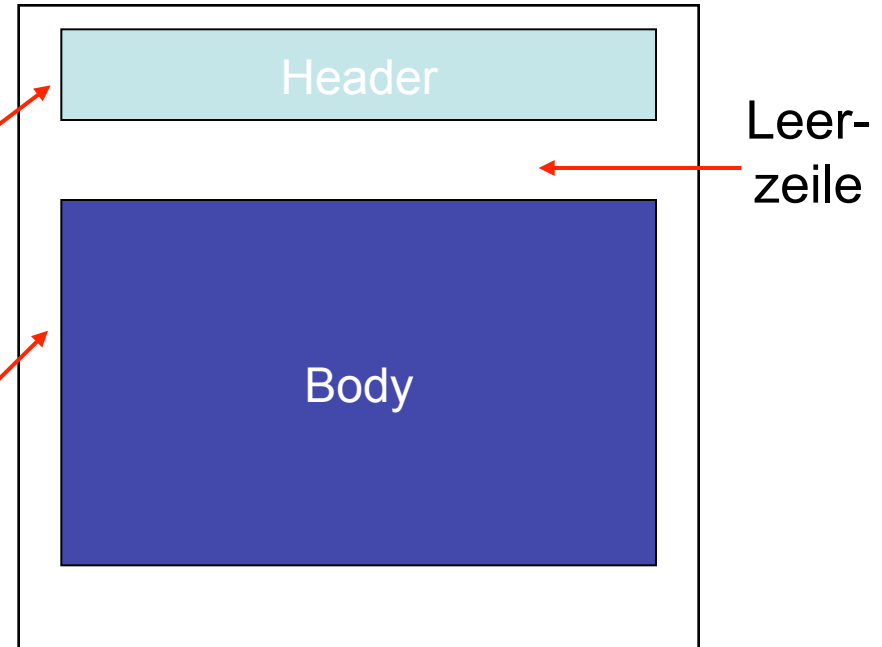
- HTTP: Pull
- SMTP: Push
- Beide kommunizieren mit ASCII-Befehl/Antwort-Paaren sowie Statuscodes
- HTTP: Jedes Objekt ist in einer eigenen Antwortnachricht gekapselt
- SMTP: Mehrere Objekte können in einer Nachricht (multipart msg) versendet werden

Format einer E-Mail-Nachricht

SMTP: Protokoll für den Austausch von E-Mail-Nachrichten

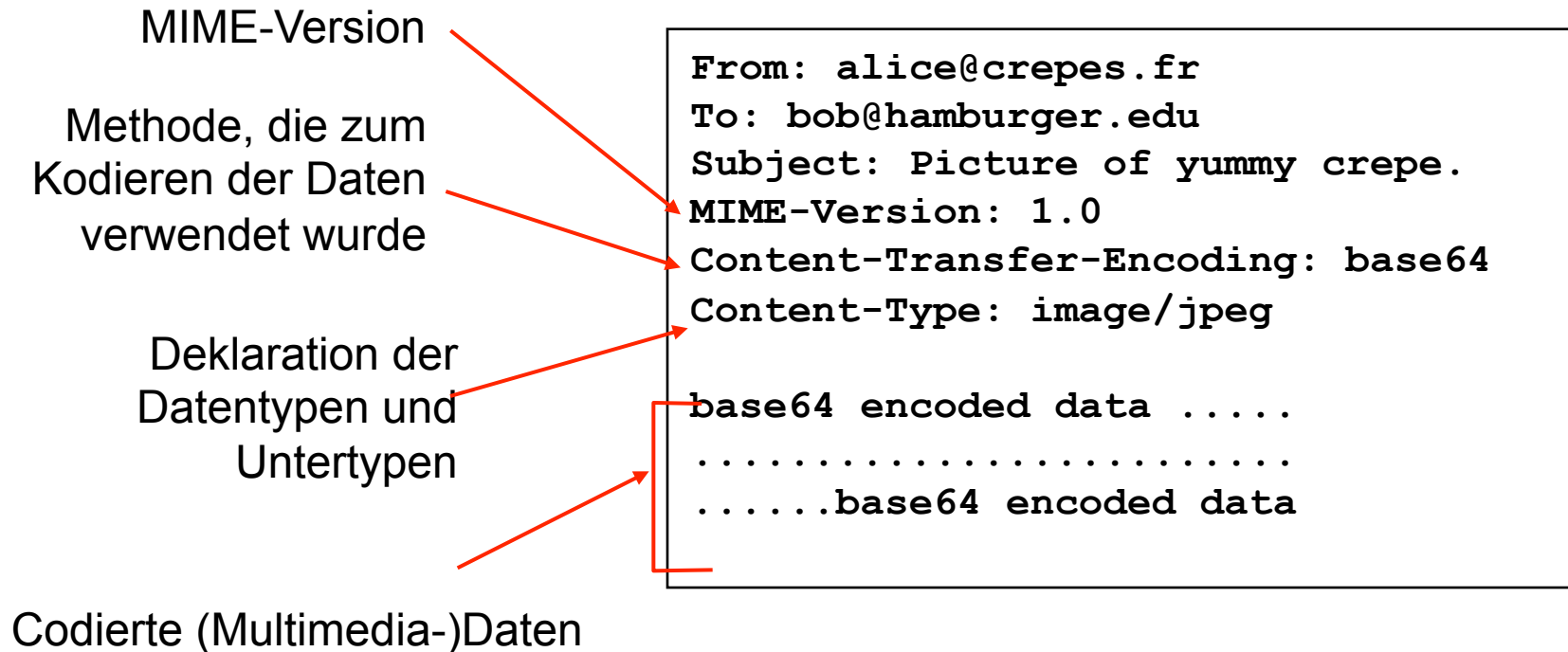
RFC 822: Standard für Textnachrichten:

- Header-Zeilen, z.B.
 - To:
 - From:
 - Subject:*Keine SMTP-Befehle!*
- Body
 - Die eigentliche Nachricht in ASCII



Nachrichtenformat: Multimedia-Erweiterung

- MIME: Multipurpose Internet Mail Extensions, RFC 2045-2047,2049,...
- Zusätzliche Zeilen im Header deklarieren den MIME-Typ des Inhaltes



MIME → Internet Medientypen

Text

Beispiele für Subtypen:

- `text/plain`
- `text/html`

Bilder

Beispiele für Subtypen:

- `image/jpeg`
- `image/gif`

Audio

Beispiele für Subtypen:

- `audio/basic` (8-bit mu-law encoded),
- `audio/mpeg` (mp3 or other mpeg Daten)

Video

Beispiele für Subtypen:


- `video/mpeg`
- `video/quicktime`

Anwendungen

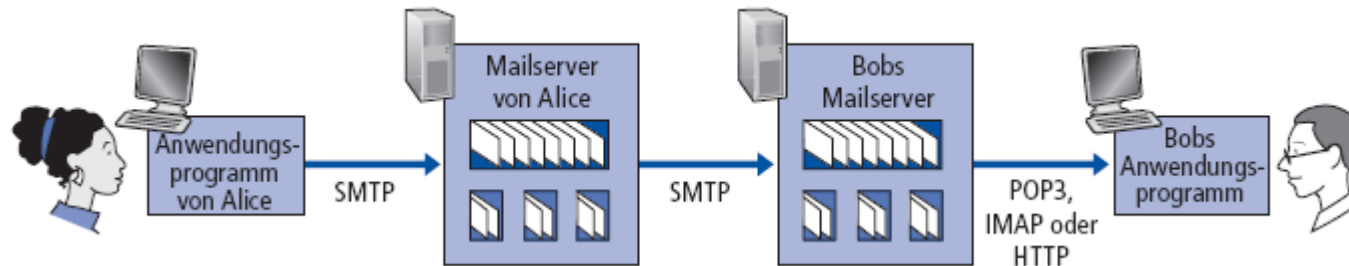
- Daten müssen von der Anwendung vor der Wiedergabe interpretiert werden
- Beispiele für Subtypen:
`application/pdf`,
`application/octet-stream`
(willkürliches binary Daten)
`application/javascript`

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=StartOfNextPart
```

```
--StartOfNextPart
Dear Bob, Please find a picture of a crepe.
--StartOfNextPart
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data .....
.....base64 encoded data
--StartOfNextPart
Do you want the recipe?
```



Mail-Zugriffsprotokolle



- SMTP: Zustellung/Speicherung auf dem Mailserver des Empfängers
- Zugriffsprotokoll: Protokolle zum Zugriff auf E-Mails
- Abruf vom Server
 - POP: Post Office Protocol [RFC 1939]
 - Autorisierung (Anwendung <--> Server) und Zugriff/Download
 - IMAP: Internet Mail Access Protocol [RFC 1730]
 - Größere Funktionalität (deutlich komplexer)
 - Manipulation der auf dem Server gespeicherten Nachrichten
 - HTTP: Hotmail, Yahoo!Mail etc.

POP3-Protokoll

Autorisierungsphase:

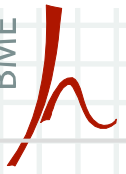
- ❑ Befehle des Clients:
 - ❖ **user**: Benutzername
 - ❖ **pass**: Passwort
- ❑ Antworten des Servers:
 - ❖ **+OK**
 - ❖ **-ERR**

Transaktionsphase:

- ❑ **list**: Nachrichten auflisten
- ❑ **retr**: Nachrichten herunterladen
- ❑ **dele**: Löschen von Nachrichten
- ❑ **Quit**: Ende

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
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 POP3 server signing off
```



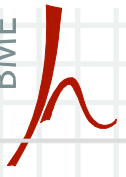
POP3 und IMAP

Mehr zu POP3

- Vorheriges Beispiel nutzte den “*Download-and-Delete*”-Modus, d.h., andere E-Mail-Clients haben danach keine Möglichkeit mehr, die Mails zu lesen
- Der “*Download-and-Keep*”-Modus ermöglicht den reinen Lesezugriff auf Nachrichten, d.h., verschiedene Clients haben Zugriff
- POP3 ist zustandslos zwischen einzelnen Sitzungen

IMAP

- Alle Nachrichten bleiben an einem Ort: auf dem Server
- Nachrichten können auf dem Server in Ordnern verwaltet werden
- IMAP bewahrt den Zustand zwischen einzelnen Sitzungen:
 - Namen von Ordnern und Zuordnung von Nachrichtennummer und Ordnername bleiben erhalten



Kapitel 2: Anwendungsschicht

- 2.1 Grundlagen
- 2.2 Web und HTTP
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS

DNS: Domain Name System

Menschen: verschiedene Identifikationsmechanismen

- Name, Ausweisnummer

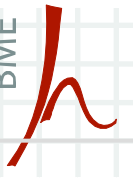
Internet-Hosts, Router:

- IP-Adresse (32 Bit) – für die Adressierung in Paketen
- “Name”, z.B.,
www.yahoo.com – von Menschen verwendet

Frage: Wie findet die Abbildung zwischen IP-Adressen und Namen statt?

Domain Name System:

- *Verteilte Datenbank*, implementiert eine Hierarchie von *Nameservern*
- *Protokoll der Anwendungsschicht*, wird von Hosts verwendet, um Namen *aufzulösen* (Abbildung zwischen Adresse und Name)
 - zentrale Internetfunktion, implementiert als Protokoll der Anwendungsschicht
 - Grund: Komplexität nur am Rand des Netzwerkes!



DNS-Dienste

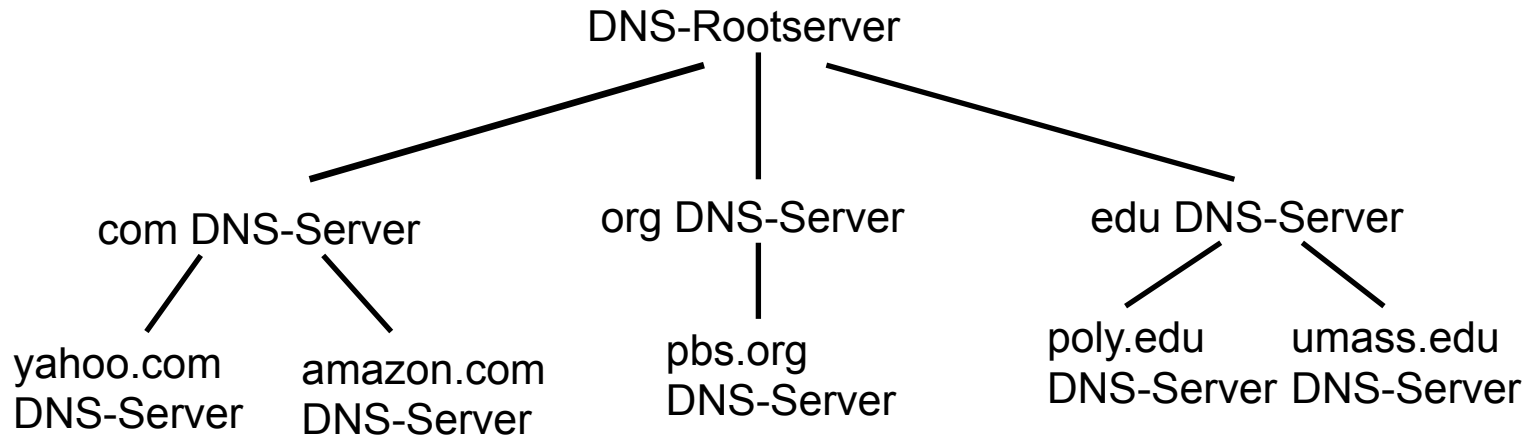
- Übersetzung von Hostnamen in IP-Adressen
- Aliasnamen für Hosts
 - Kanonische Namen und Aliasnamen
- Aliasnamen für Mailserver
- Lastausgleich
 - Replizierte Webserver: mehrere IP-Adressen von einem kanonischen Namen

Warum ist DNS nicht zentralisiert?

- Robustheit gegenüber Fehlern und Angriffen
- Datenverkehrsmenge
- Große „Distanz“ zur zentralisierten Datenbank
- Wartung

Skaliert nicht!

Verteilte, hierarchische Datenbank



Client sucht die IP-Adresse von www.amazon.com – erste

Annäherung:

- Client fragt seinen lokalen DNS-Server
- Dieser fragt einen DNS-Rootserver, um den DNS-Server für com zu finden
- Danach fragt er den com-DNS-Server, um den amazon.com-DNS-Server zu finden
- Dann wird der amazon.com-DNS-Server gefragt, um die IP-Adresse zu www.amazon.com zu erhalten

DNS: Root-Nameserver

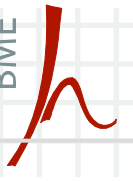
- Wird vom lokalen Nameserver kontaktiert, wenn dieser einen Namen nicht auflösen kann
- Root-Nameserver:
 - Kennt die Adressen der Nameserver der Top-Level-Domains (com, net, org, de, uk, ...)
 - Gibt diese Informationen an die lokalen Nameserver weiter



13 Root-Nameserver
weltweit

TLD- und Autoritative Server

- **Top-Level-Domain (TLD)-Server:**
 - Verantwortlich für com, org, net, edu etc. sowie für alle Länder-Domains, z.B. de, uk, fr, ca, jp
 - Verisign Inc. ist verantwortlich für den com-TLD-Server
 - Internet Szolgálatok Tanácsa (NIC.hu) hat die Verantwortung für den hu-TLD-Server
- **Autoritativer DNS-Server:**
 - DNS-Server einer Organisation, der eine autorisierte Abbildung der Namen dieser Organisation auf IP-Adressen anbietet
 - Verwaltet von der entsprechenden Organisation oder einem Service Provider



Lokale Nameserver

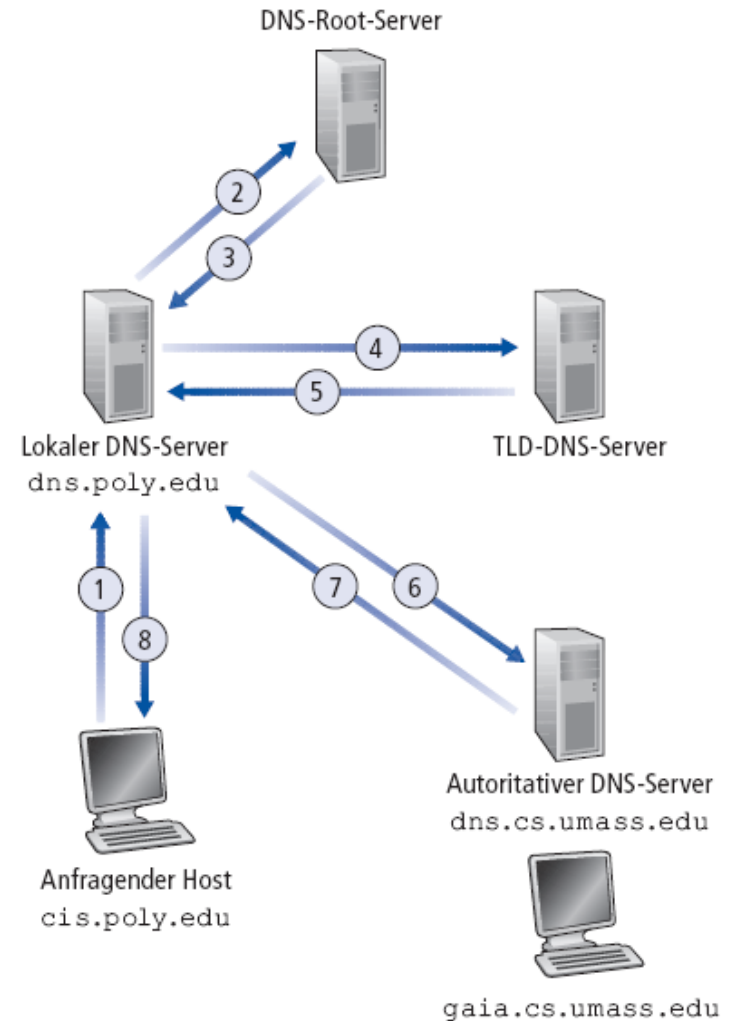
- Gehört nicht zur Hierarchie der DNS-Server
- Jeder ISP (ISP für Privatkunden, Firmen, Universität) besitzt einen lokalen Nameserver
 - Werden auch “Default-Nameserver” genannt
- Wenn ein Host eine DNS-Anfrage startet, dann schickt er diese an seinen lokalen Nameserver
 - Dieser kümmert sich um die Anfrage so lange, bis eine endgültige Antwort vorliegt
 - Dazu kontaktiert er bei Bedarf Root-Nameserver, TLD-Nameserver und autoritative Nameserver
 - Dann schickt er die Antwort an den Host zurück

Iterative Namensauflösung mit DNS

- Host cis.poly.edu fragt nach der IP-Adresse von gaia.cs.umass.edu

Iteratives Vorgehen:

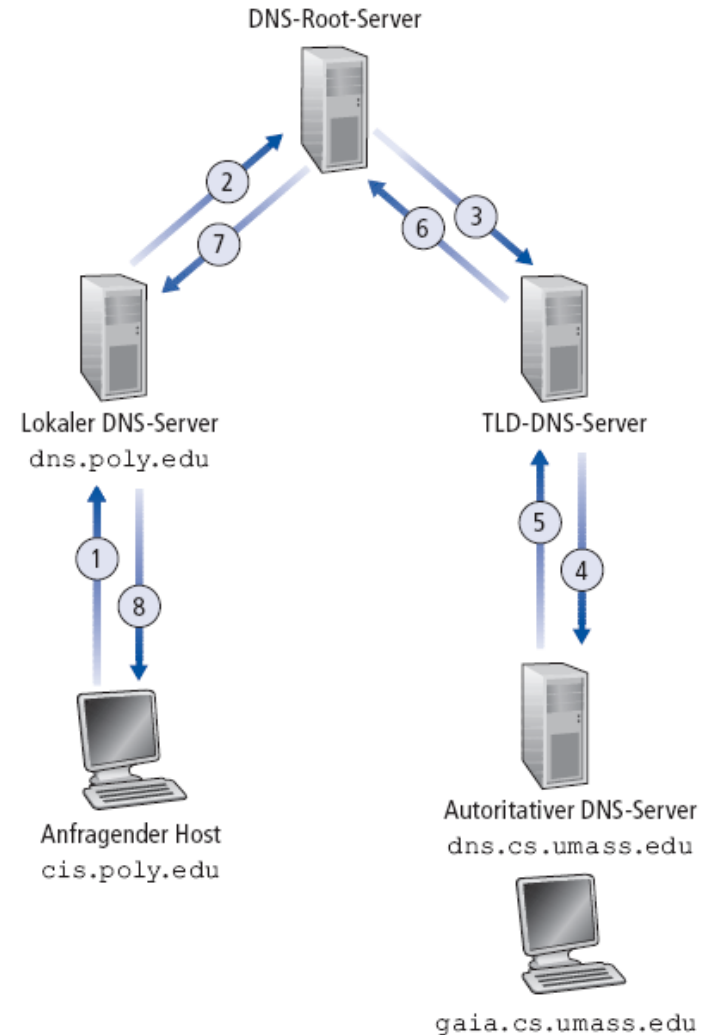
- Angesprochene Server in der Hierarchie antworten mit einem Verweis auf andere Server
- “Ich kenne den Namen nicht, frag’ diesen Server ...”



Rekursive Namensauflösung mit DNS

Rekursives Vorgehen:

- Die Aufgabe zur Namensauflösung wird an den gefragten Nameserver delegiert
- Zusätzliche Belastung!
- Root-Nameserver erlauben dies häufig nicht
- Andere Nameserver dagegen schon!
- Vorteil: Caching der Antworten



- Sobald ein Nameserver eine Abbildung zur Namensauflösung kennenlernt, merkt er sich diesen in einem Cache
 - Die Einträge im Cache werden nach einer vorgegebenen Zeit wieder gelöscht
 - Die Adressen der TLD-Server werden üblicherweise von den lokalen Nameservern gecacht
 - Root-Nameserver werden eher selten angesprochen
- Mechanismen zur Pflege von Cache-Einträgen und zur Benachrichtigung bei Änderungen werden derzeit von der IETF entwickelt
 - RFC 2136
 - <http://www.ietf.org/html.charters/dnsind-charter.html>

DNS Resource Records

DNS: Verteilte Datenbank für Resource Records (RR)

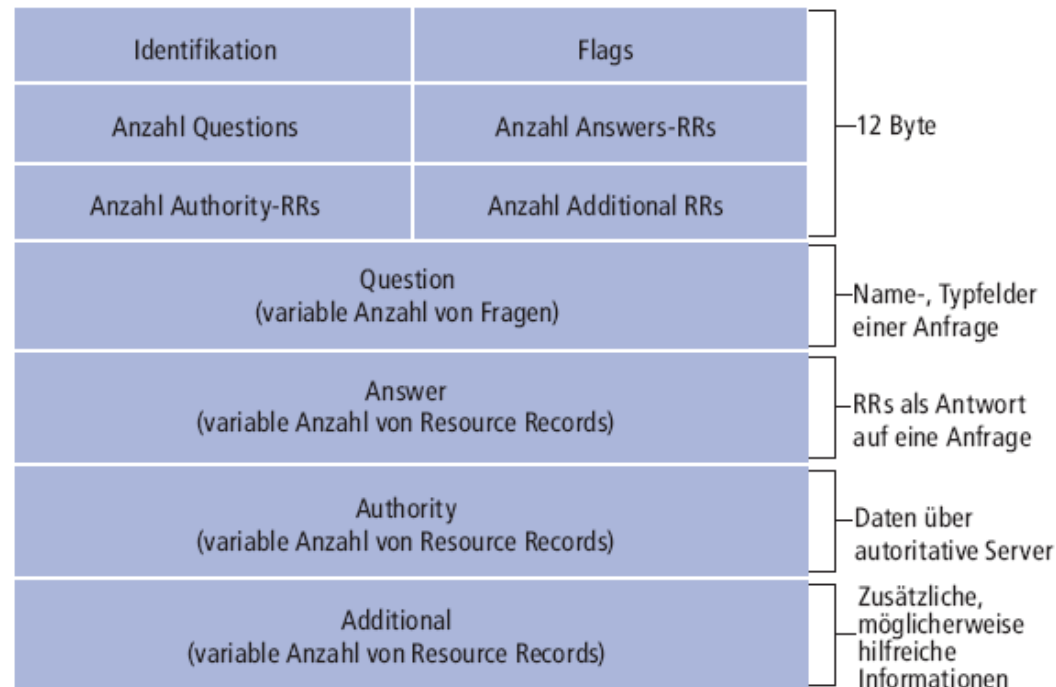
RR-Format: (**name**, **wert**, **typ**, **ttl**)

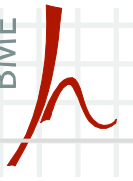
- Typ=A
 - **name** ist der Hostname
 - **value** ist die IP-Adresse
- Typ=NS
 - **name** ist eine Domain (z.B. foo.com)
 - **value** ist der Hostname des autoritativen Nameservers für diese Domain
- Typ=CNAME
 - **name** ist ein Alias für einen kanonischen (echten) Namen:
 - `www.ibm.com` ist ein Alias für `servereast.backup2.ibm.com`
 - **value** ist der kanonische Name
- Type=MX
 - **name** ist eine Domain (z.B. foo.com)
 - **value** ist der Name des Mailservers für die Domain

DNS-Protokoll: *Query*- und *Reply*-Nachrichten, beide mit demselben Nachrichtenformat

Header-Felder

- *identification*: 16-Bit-ID, wird für die Query-Nachricht vergeben, die Reply-Nachricht verwendet dieselbe ID
- *flags*:
 - query/reply
 - recursion desired
 - recursion available
 - reply is authoritative





Neue Einträge in DNS einfügen

- Beispiel: neues Startup “Network Utopia”
- Registrieren des Namens networkutopia.com bei einem *DNS-Registrar* (z.B. Network Solutions)
 - Bereitstellen der Namen und der IP-Adressen der autoritativen Server (Primary und Secondary)
 - Registrar trägt zwei RRs beim TLD-Server für com ein:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

- Wie könnte der Typ-A RR für `www.networkutopia.com` aussehen? Wie der Typ-MX-RR für `networkutopia.com`?
- Wie erfährt man die IP-Adresse Ihrer Webseite?

DNS Beispiel

```

; <<>> DiG 9.6-ESV-R4-P3 <<>> www.google.com +all
;; global options: +cmd
;; mark@ip10-105-71:~$ dig www.google.com NS +all
;;
;; ; <<>> DiG 9.6-ESV-R4-P3 <<>> www.google.com NS +all
;;
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12050
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; ;; QUESTION SECTION:
www ;www.google.com.                IN      NS

www
www ;; AUTHORITY SECTION:
www google.com.                30      IN      SOA      ns1.google.com. dns-admin.google.c

www
;; Query time: 11 msec
;; SERVER: 10.105.1.254#53(10.105.1.254)
;; WHEN: Tue Feb 26 11:27:12 2013
;; MSG SIZE rcvd: 82

```