Datenkommunikation

Anwendungsschicht

Fallstudien DNS, HTTP, E-Mail,...

Wintersemester 2011/2012

Überblick

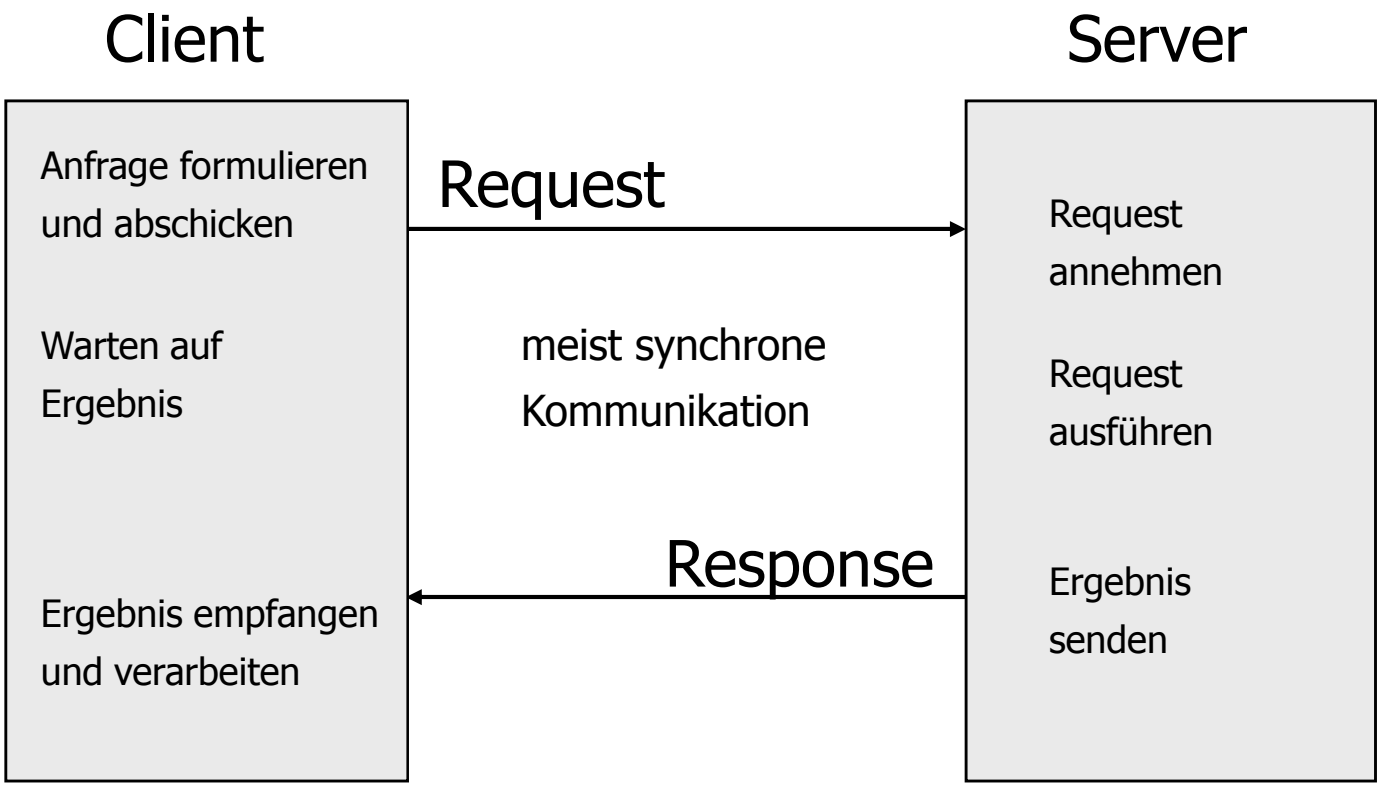
1	Grundlagen von Rechnernetzen, Teil 1
2	Grundlagen von Rechnernetzen, Teil 2
3	Transportzugriff
4	Transportschicht, Grundlagen
5	Transportschicht, TCP (1)
6	Transportschicht, TCP (2) und UDP
7	Vermittlungsschicht, Grundlagen
8	Vermittlungsschicht, Internet
9	Vermittlungsschicht, Routing
10	Vermittlungsschicht, Steuerprotokolle und IPv6
11	Anwendungsschicht, Fallstudien
12	Mobile IP und TCP

Überblick

1. **Client-/Server versus Peer-to-Peer (P2P)**
2. Domain Name System
3. HTTP
4. E-Mail
5. Multimedia-Protokolle

Client-/Server-Kommunikation

- Klare Rollenaufteilung

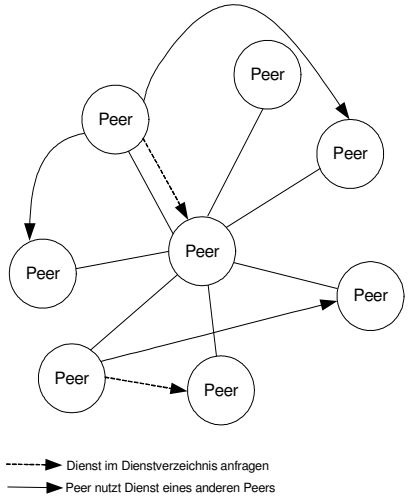


Peer-to-Peer-Architekturen (1)

- Gleichberechtigte Peers sind in der Rolle von Server und Client (meist im Internet)
- Beispiele für P2P-Systeme/-Protokolle:
  - **BitTorrent**: Filesharing-System im Internet
  - **Skype**: VoIP-System
  - **Bitcoin**: Digitales Cash-System
  - **Napster**: Austausch von Musik-Dateien (rechtl. Probleme)
  - **Gnutella**: Filesharing-Netzwerk im Internet (nicht mehr relevant)
  - **KaZaa**: Internet-Tauschbörse für Musikdateien, Videos, Textdokumente und Bilder

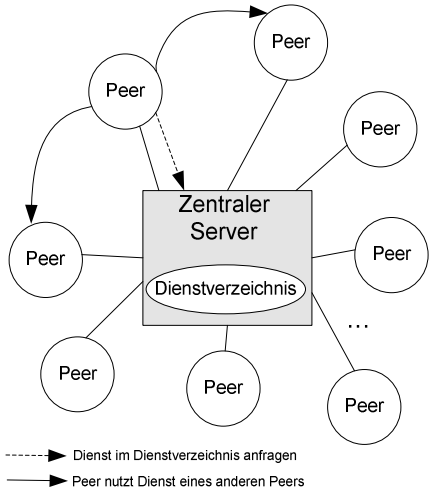
Peer-to-Peer-Architekturen (2)

- Verschiedene Varianten: pur, hybrid, Superpeer



Pur

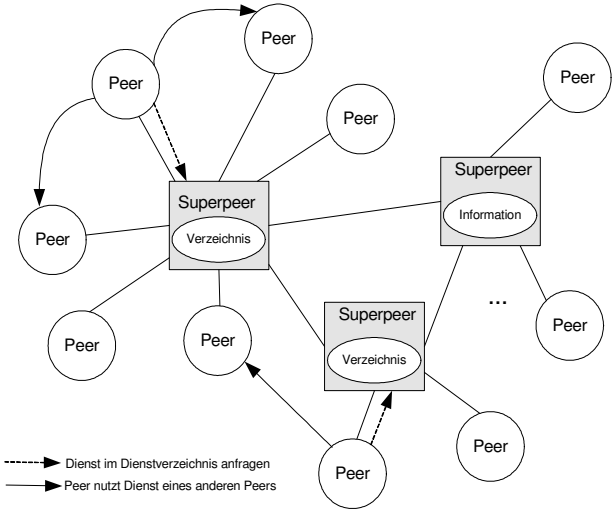
Beispiel: Gnutella



Hybrid

Beispiele: Napster

Skype, BitTorrent



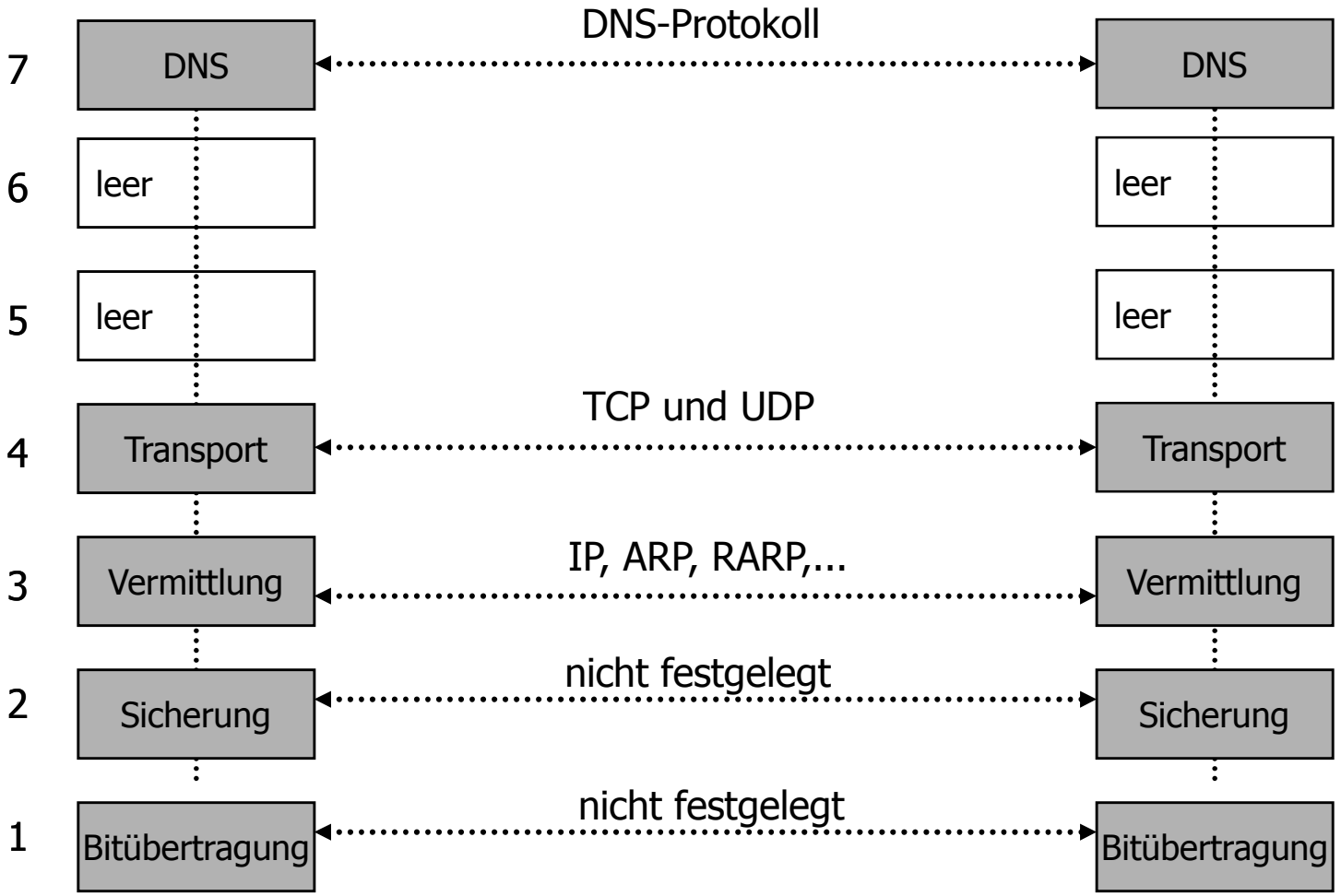
Superpeer:

Beispiel: KaZaa

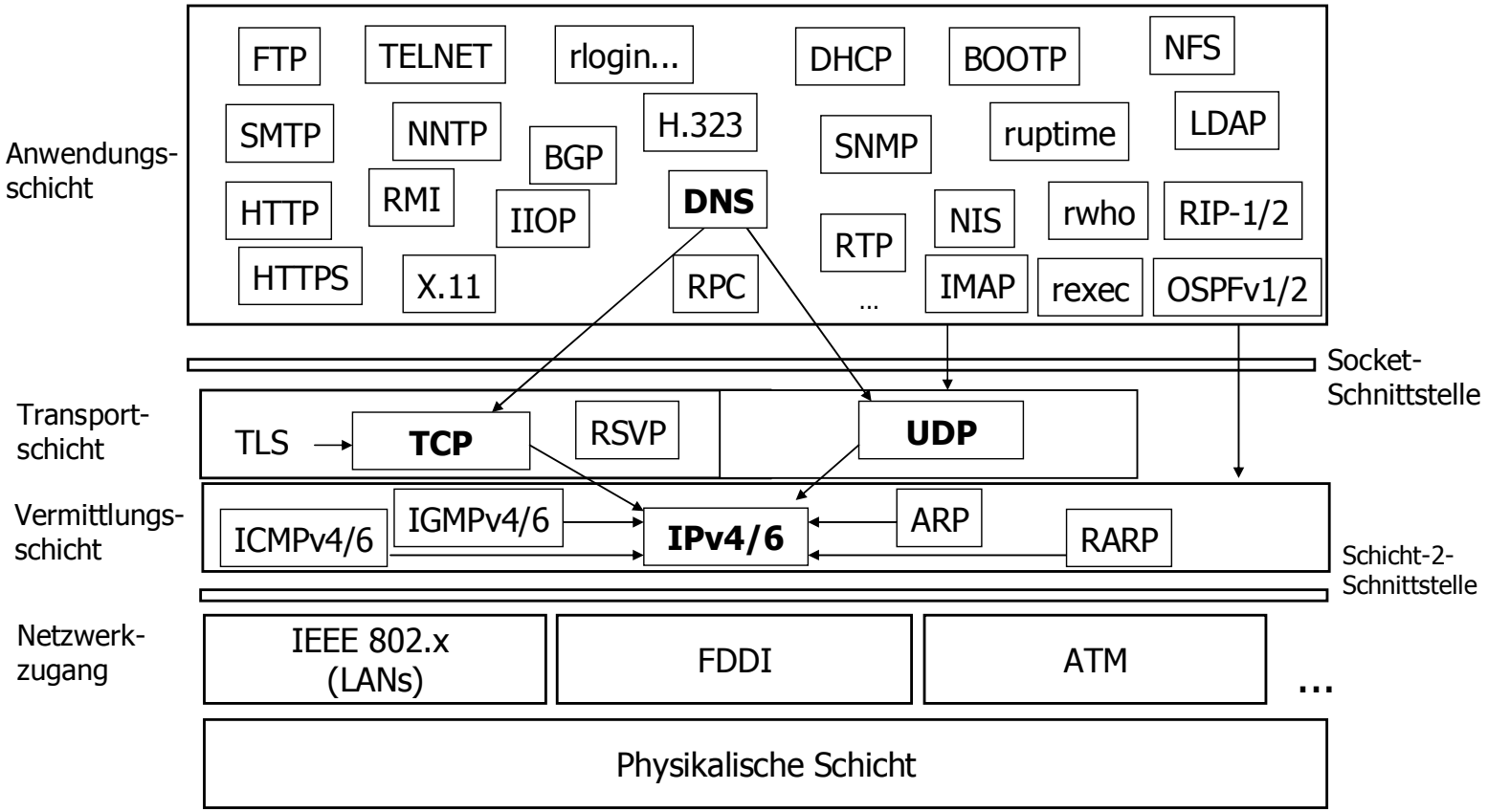
Überblick

1. Client-/Server versus Peer-to-Peer (P2P)
2. **Domain Name System**
3. HTTP
4. E-Mail
5. Multimedia-Protokolle

Überblick
Einordnung in die TCP/IP-Protokollfamilie



Übersicht
Einordnung in die TCP/IP-Protokollfamilie



DNS-Port: 53 TCP/UDP

Domain Name System (DNS): Hintergrund

- Im ARPANET mit einigen hundert Hosts war die Verwaltung der Adressen noch einfach
  - Es gab eine Datei **hosts.txt** auf einem Verwaltungsrechner, die alle IP-Adressen enthielt (flacher Namensraum)
  - Die Datei wurde nachts auf die anderen Hosts kopiert
- Als das Netz immer größer wurde, führte man DNS ein
- DNS dient prinzipiell der **Abbildung von Hostnamen auf eMail- und IP-Adressen**
- DNS ist ein Internet **Directory Service**

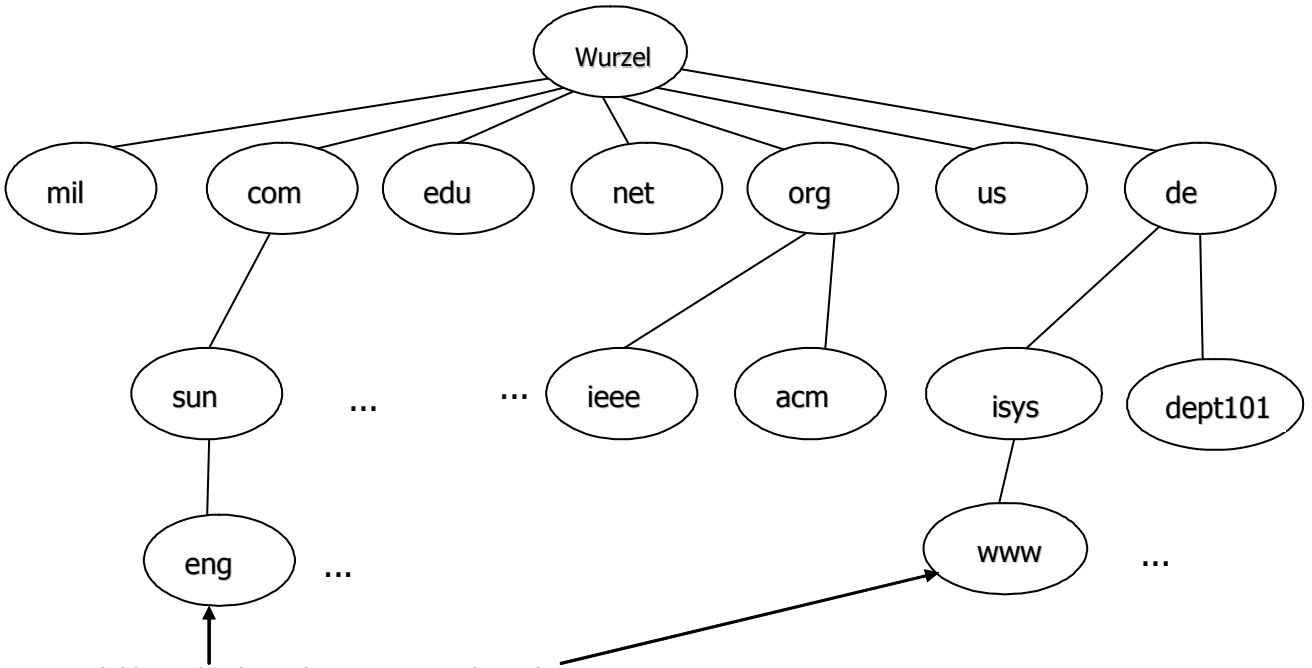
Achtung: DNS nicht mit Routing verwechseln!

Domänen und Subdomänen

- DNS ist in den RFCs 1034 und 1035 definiert
- DNS ist ein hierarchischer Namensverzeichnis für IP-Adressen (Adressbuch des Internets)
- DNS verwaltet eine **Datenbank**, die sich über zahlreiche Internet-Hosts erstreckt
- Konzeptionell ist das Internet in **mehrere hundert Domänen** aufgeteilt
- Die Domänen sind wiederum in **Teildomänen (Subdomains)** untergliedert, usw.

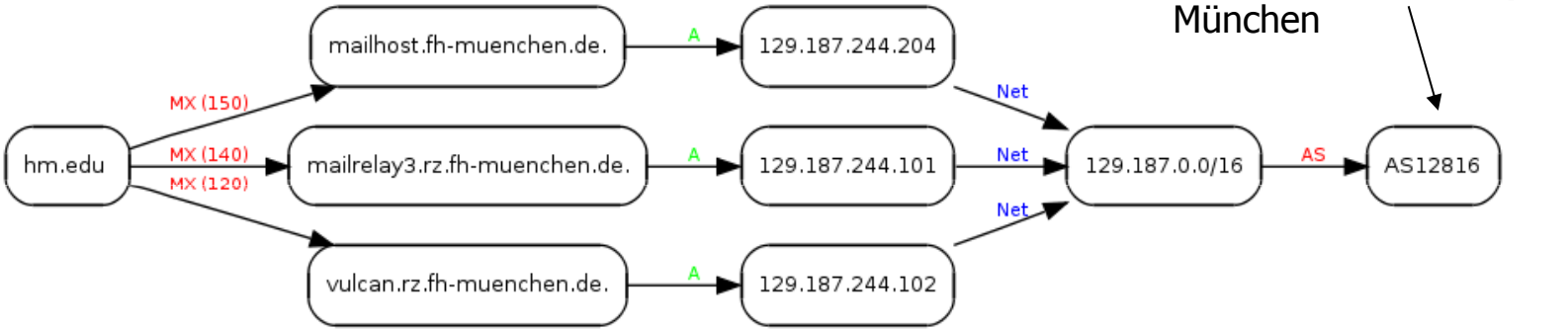
DNS-Namensraum

- Weltweit verteilter Namensraum
- **Baum**, an dessen Blättern dann die Hosts hängen (**rein organisatorisch**, nicht physikalisch)

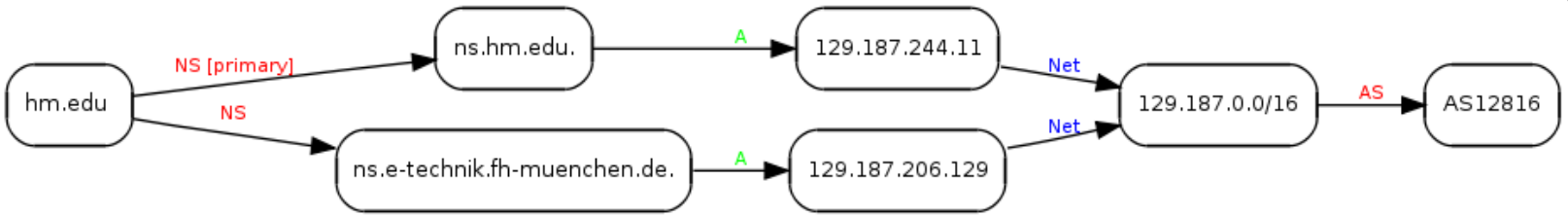


Einschub: Suche nach Domain-Information (1)

- Beispieldienst über http://serversniff.net (oder .de)
- NS-Report für hm.edu ergibt:
  - Mail-Networkstructure



  - Nameserver-Networkstructure



Einschub: NS-Report für hm.edu (Auszug)

NS-Record(s) for domain hm.edu:
ns.fh-muenchen.de. 129.187.244.11
ns.e-technik.fh-muenchen.de. 129.187.206.129

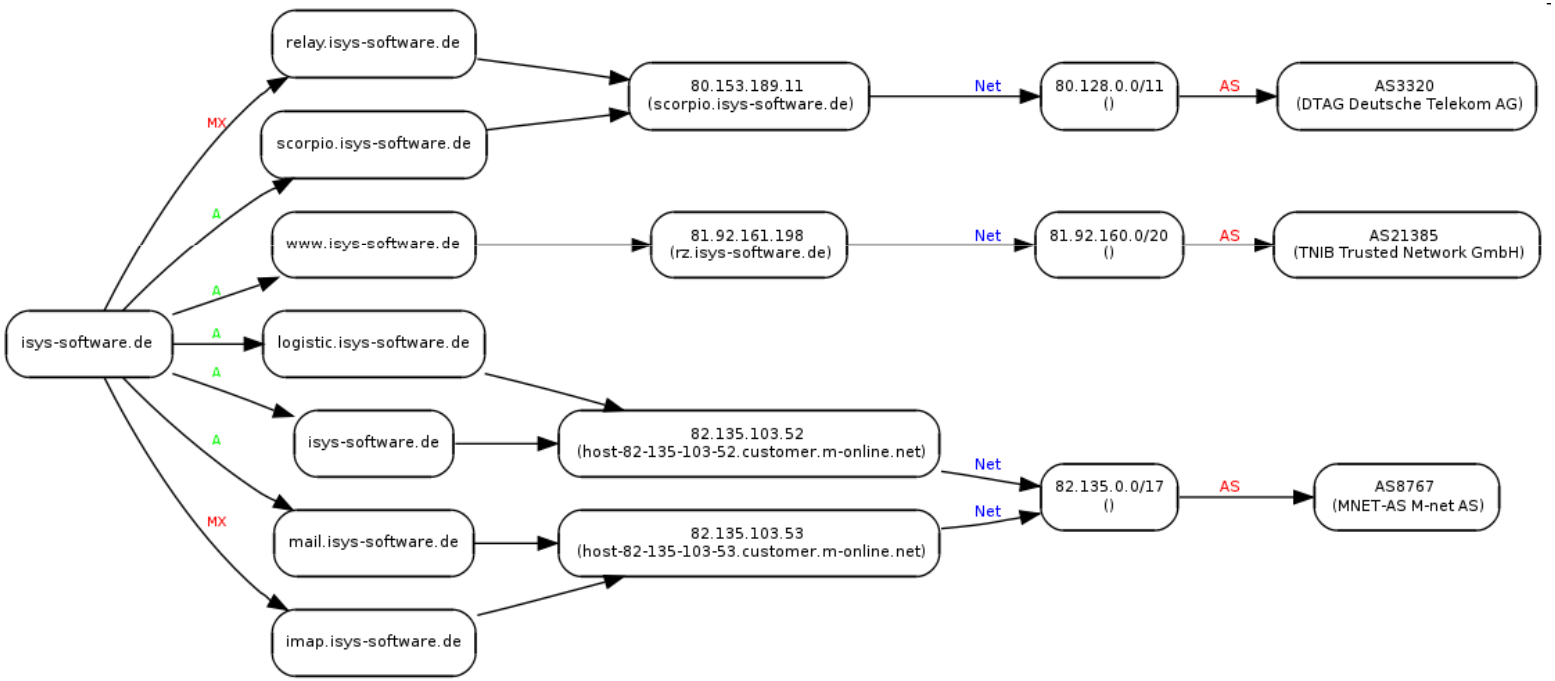
MX-Servers for hm.edu:
150 mailhost.fh-muenchen.de. 129.187.244.204
140 mailrelay3.rz.fh-muenchen.de. 129.187.244.101
120 vulcan.rz.fh-muenchen.de. 129.187.244.102

SOA-Record for hm.edu:
ns.hm.edu. hostmaster.hm.edu. 2009121462 10800 1800 3600000 86400
Serial: 2009121462
Refresh: 10800
Retry: 1800
Expire: 3600000 (1000 hours or 42 days)
TTL: 86400

Recursive-Queries:
ns.e-technik.fh-muenchen.de. YES - recursive queries allowed!
ns.hm.edu. YES - recursive queries allowed!
...

Einschub: Suche nach Domain-Information (2)

- Beispieldienst über http://serversniff.de
- Domainreport für isys-software.de ergibt:



Weitere info zur Domäne über

http://who.is/whois-de/ip-address/isys-software.de/

Nicht gesponserte Domains

- **ICANN** (Internet Corporation for Assigned Names and Numbers) pflegt diese „nicht-gesponserten“ Domains
- Man unterscheidet:
  - Geographische oder länderspezifische (**Country-Code**, **ccTLDs**)

    Domains wie de, at, us, uk, gb, usw.
    - Für jedes Land ist nach ISO 3166 ein Code mit zwei Buchstaben vorgesehen.
    - Es gibt derzeit über 200 ccTLDs.
    - Für die Europäische Union wurde *.eu* als Gemeinschaft ebenfalls dieser Art von Domains zugeordnet, obwohl *.eu* als eine Ausnahme behandelt wird (Liste der Ausnahmen zu ISO 3166).
  - Allgemeine Domains für Organisationen (**generic** oder **gTLDs**)
  - **Infrastruktur-Domains** als Sonderfall (spezielle Domain .arpa)
- Es gibt darüber hinaus „gesponserte“ Domains

Generic TLDs

Domain	**Beschreibung**
com	Kommerzielle Organisationen (sun.com, ibm.com, ...)
edu	Bildungseinrichtungen (fhm.edu)
gov	Amerikanische Regierungsstellen (nfs.gov)
mil	Militärische Einrichtung in den USA (navy.mil)
net	Netzwerkorganisationen (nsf.net)
org	Nichtkommerzielle Organisationen
int	Internationale Organisationen (nato.int)
bis	Business, für Unternehmen
info	Informationsanbieter
arpa	TLD des ursprünglichen Arpanets, die heute als sog. *Address and Routing Parameter Area* verwendet wird (auch als "Infrastruktur-Domain" bezeichnet).
pro	Professions, Berufsgruppen der USA, Deutschland und des Vereinigten Königreichs

Gesponserte Domains

- Werden von unabhängigen Organisationen in Eigenregie kontrolliert und finanziert
- Eigene Richtlinien für die Vergabe von Domainnamen
- Zu den „gesponserten" TLDs gehören:
  - **.aero**: Aeronautics, für in der Luftfahrt tätige Organisationen, weltweiter Einsatz
  - **.coop**: Steht für *cooperatives* (Genossenschaften), weltweiter Einsatz
  - **.info**: Informationsanbieter, weltweiter Einsatz
  - **.int**: Internationale Regierungsorganisationen (Beispiel www.nato.int oder www.eu.int)
  - **.mobi**: Darstellung von Webseiten speziell für mobile Endgeräte, weltweiter Einsatz ...

DNS-Namensraum und DNS-Datenbank

- DNS ist eine baumförmige **weltweite Vernetzung** von Name-Servern
- DNS bildet eine weltweit verteilte Datenbank (**DNS-Datenbank**)
- Jeder Knoten im DNS-Baum stellt eine Domäne dar und kann mit einem Verzeichnis in einem Dateisystem verglichen werden
- Jeder Knoten hat einen Namen

Root-Name-Server

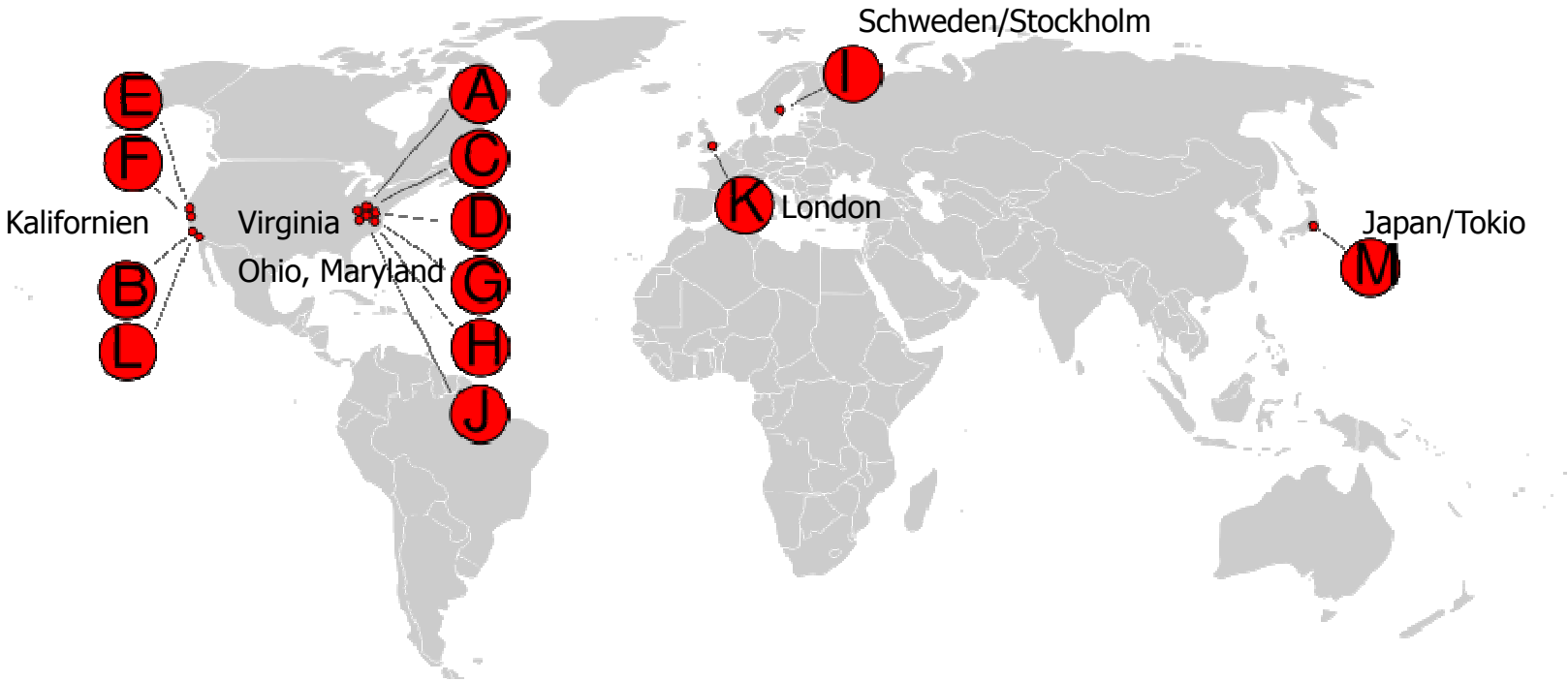
- Es gibt derzeit **weltweit 13 sog. Root-Name-Server (A...M)**
  - 10 in Nordamerika, 1 in Stockholm, 1 in London, 1 in Tokio
- Root-Name-Server geben Informationen über die weitere Suche
- Ein Root-Name-Server
  - verfügt über eine Referenz-Datenbank aller von der ICANN freigegebenen Top-Level-Domains (TLD) und
  - die wichtigsten Referenzen auf die sog. Top-Level-Domain-Server
- Ein Root-Name-Server kennt somit immer einen DNS-Server, der eine Anfrage beantworten kann

Root-Name-Server

- DNS erlaubt DNS-PDUs (UDP) bis zu einer Größe von 512 Byte
  - Daher max. 13 DNS-Records in einer PDU
  - EDNS (Extended DNS) als DNS-Erweiterung ermöglicht die zehnfache Größe (RFC 2671)
- A und J haben **zentrale** Bedeutung. Sie verteilen die Datenbasis für alle anderen Root-Name-Server zweimal täglich
- Root-Name-Server bestehen aus z.T. **vielen Einzelservern**:
  - Root-Name-Server F besteht aus 33 Serverrechnern (2009)
  - Root-Name-Server K besteht aus aus 16 Serverrechnern (2009)
- Diese Root-Name-Server sind auf der ganzen Welt verteilt und einige nutzen heute einen **Anycast-Mechanismus** (eine IP-Adresse) zur Datenverteilung

Root-Name-Server (früher)

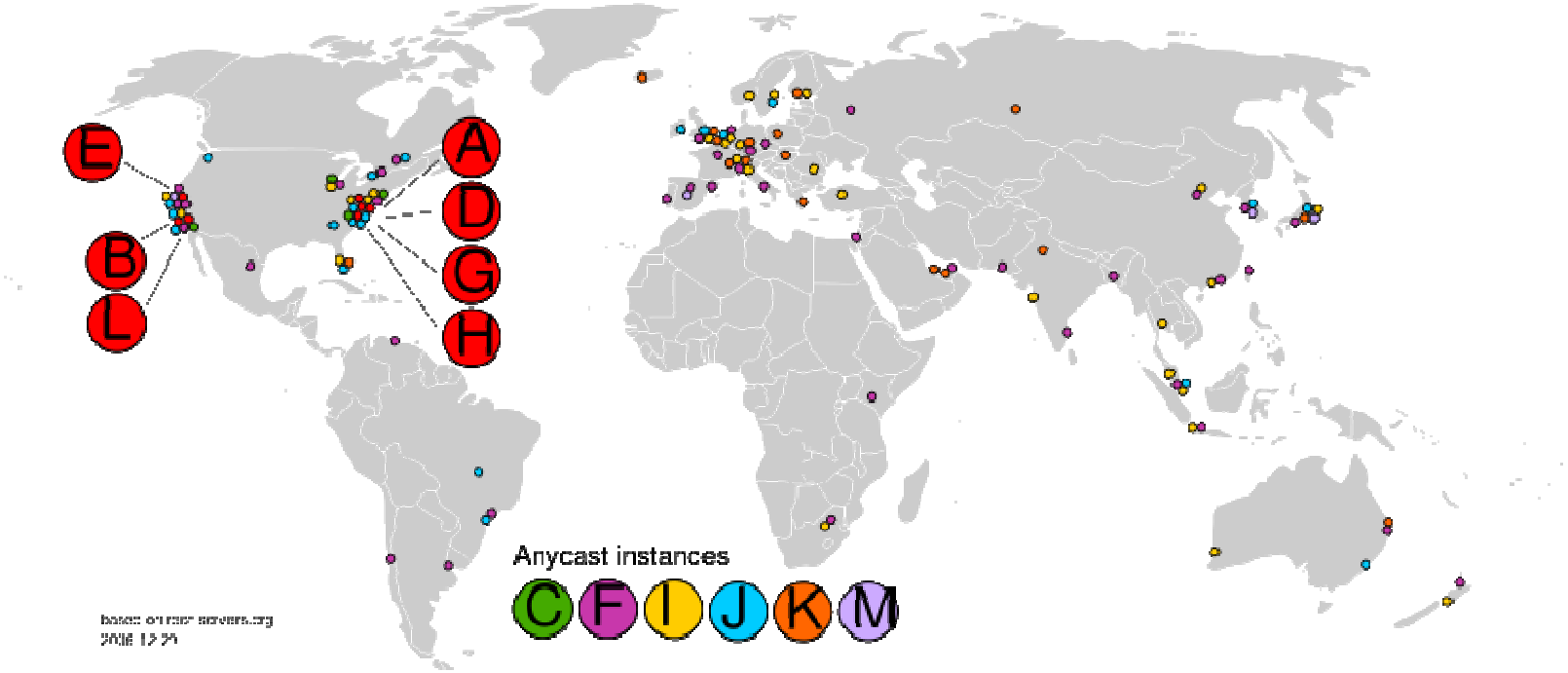
- Single Point of Failure: Synchronisation erfolgte über Root-Name-Server A (VeriSign = Amerikanisches Unternehmen und Zertifizierungsstelle für digitale Zertifikate)
- Viele DoS-Angriffe



Quelle: Wikipedia

Root-Name-Server (heute, mit Anycast)

- Mehr Ausfallsicherheit, in 2009 mehr als 120 Root-Name-Server verfügbar
- Keine Synchronisation mit A → Kein Single Point of Failure



Quelle: Wikipedia

Anycast nutzt Routing

→ nächster DNS-Server erhält den DNS-Request

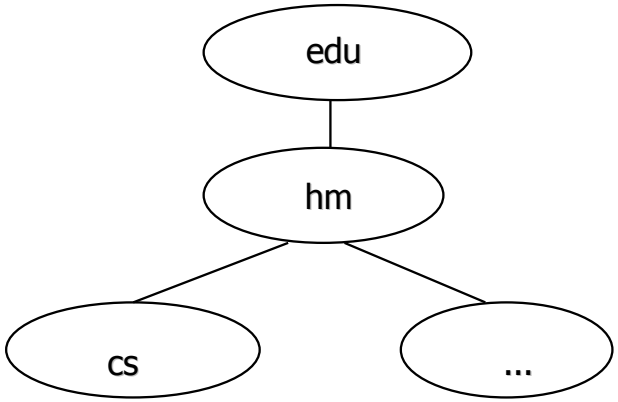
Root-Name-Server

Root-Server	Alter Name	Betreiber	Ort
A	ns.internic.net	VeriSign	Dulles, Virginia, USA
B	ns1.isi.edu	ISI	Marina Del Rey, Kalifornien, USA
C	c.psi.net	Cogent Communications	Nutzt Anycast
D	terp.umd.edu	University of Maryland	College Park, Maryland, USA
E	ns.nasa.gov	NASA	Mountain View, Kalifornien, USA
F	ns.isc.org	ISC	Nutzt Anycast
G	ns.nic.ddn.mil	U.S. DoD NIC	Columbus, Ohio, USA
H	aos.arl.army.mil	U.S. Army Research Lab	Aberdeen Proving Ground, Maryland, USA
...			

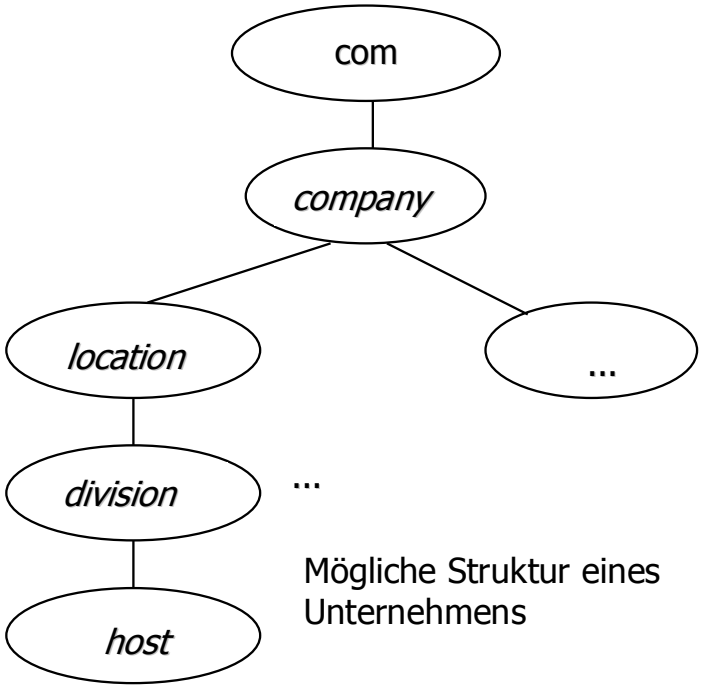
Quelle: Wikipedia

DNS-Namensraum

- **Beispiele** für Domänen mit untergeordneten Subdomänen



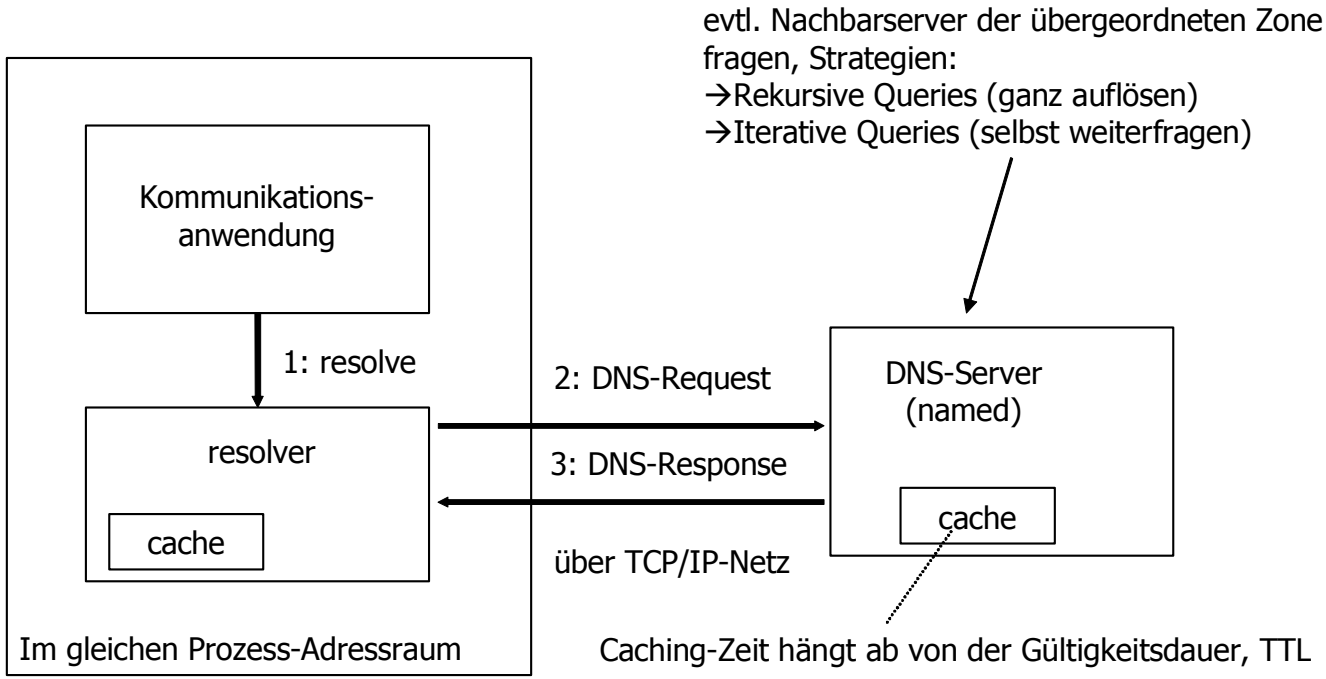
Hochschule München



Mögliche Struktur eines Unternehmens

Adressauflösung

- Eine Anwendung, die eine Adresse benötigt, wendet sich lokal an einen **Resolver** (Library), der eine Anfrage an einen DNS-Server (unter Unix z.B. **named**) richtet.



Name Server und Zonen

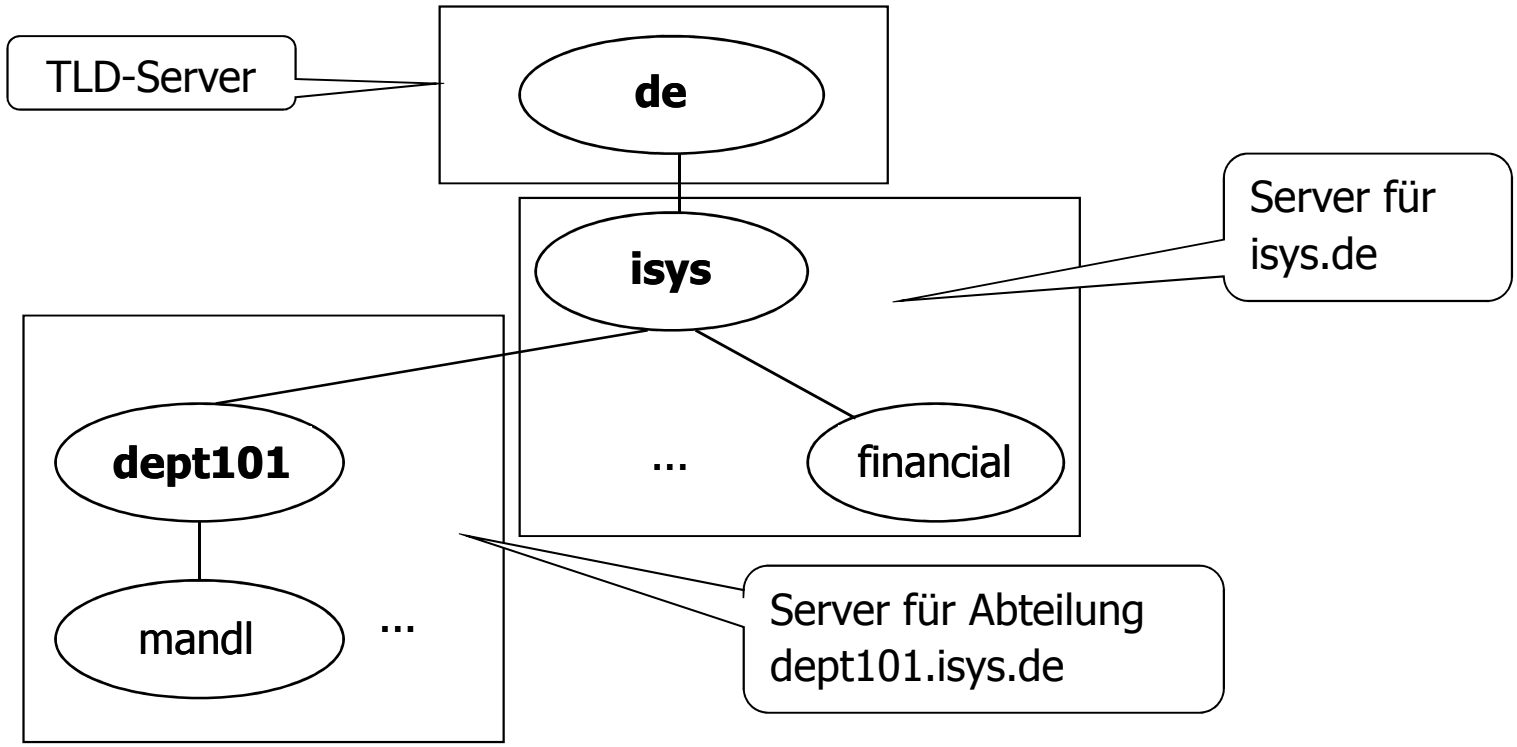
- DNS-Name-Server verwaltet jeweils **Zonen** des DNS-Baums, wobei eine Zone an einem Baumknoten beginnt und die darunter liegenden Zweige beinhaltet
- Ein Name-Server (bzw. die entspr. Organisation) kann die Verantwortung für Subzonen an einen weiteren Name Server **delegieren**
- Die Name-Server kennen jeweils ihre Nachbarn in der darunter- oder darüberliegenden Zone
- Informationen des DNS werden in sog. **Resource Records** verwaltet

Autoritative Name Server und andere...

- **Autoritativer** Name Server:
  - Verantwortlich für eine Zone
  - Mind. einer muss in Zone sein (Primary Nameserver)
  - Kennt alle Adressen der Zone genau
  - Siehe Konfigurationsdatei (**SOA-Resource-Record**)
- **Nicht-autoritativer** Name Server:
  - Bezieht Informationen von anderen Servern
  - Verfalldatum über TTL-Parameter geregelt
- **Zonentransfer** zwischen Primary und Secondary Server

Name Server und Zonen

- Beispiel für die Delegation:
  - Root-Name-Server kennt isys-Server
  - isys-Server kennt dept101-Server



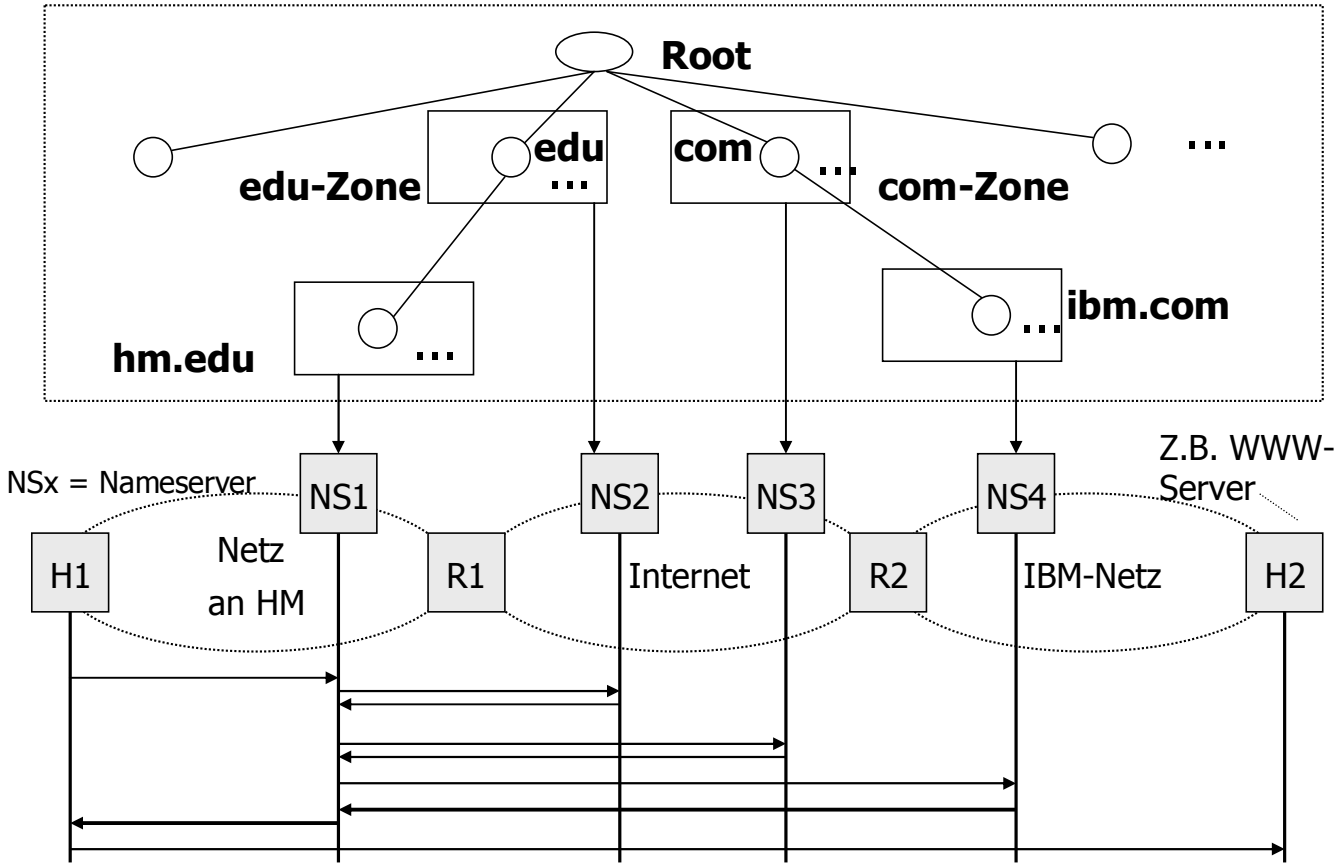
Name Server

- In jeder Zone muss es mindestens zwei Name Server geben (**primary und secondary**), die auch miteinander kommunizieren
- Für normale Anfragen wird ein **UDP-basiertes** Protokoll verwendet, größere Abfragen über TCP
- Primary und Secondary kommunizieren über **TCP**
- Name Server werden von Internet Service Providern (ISP) und Netzbetreibern betrieben
- Die Domain **.de** verwaltet das Deutsche Network Information Center (**DENIC**)
  - betrieben in Frankfurt (früher TU Dortmund, dann TU Karlsruhe)

Domainnamen

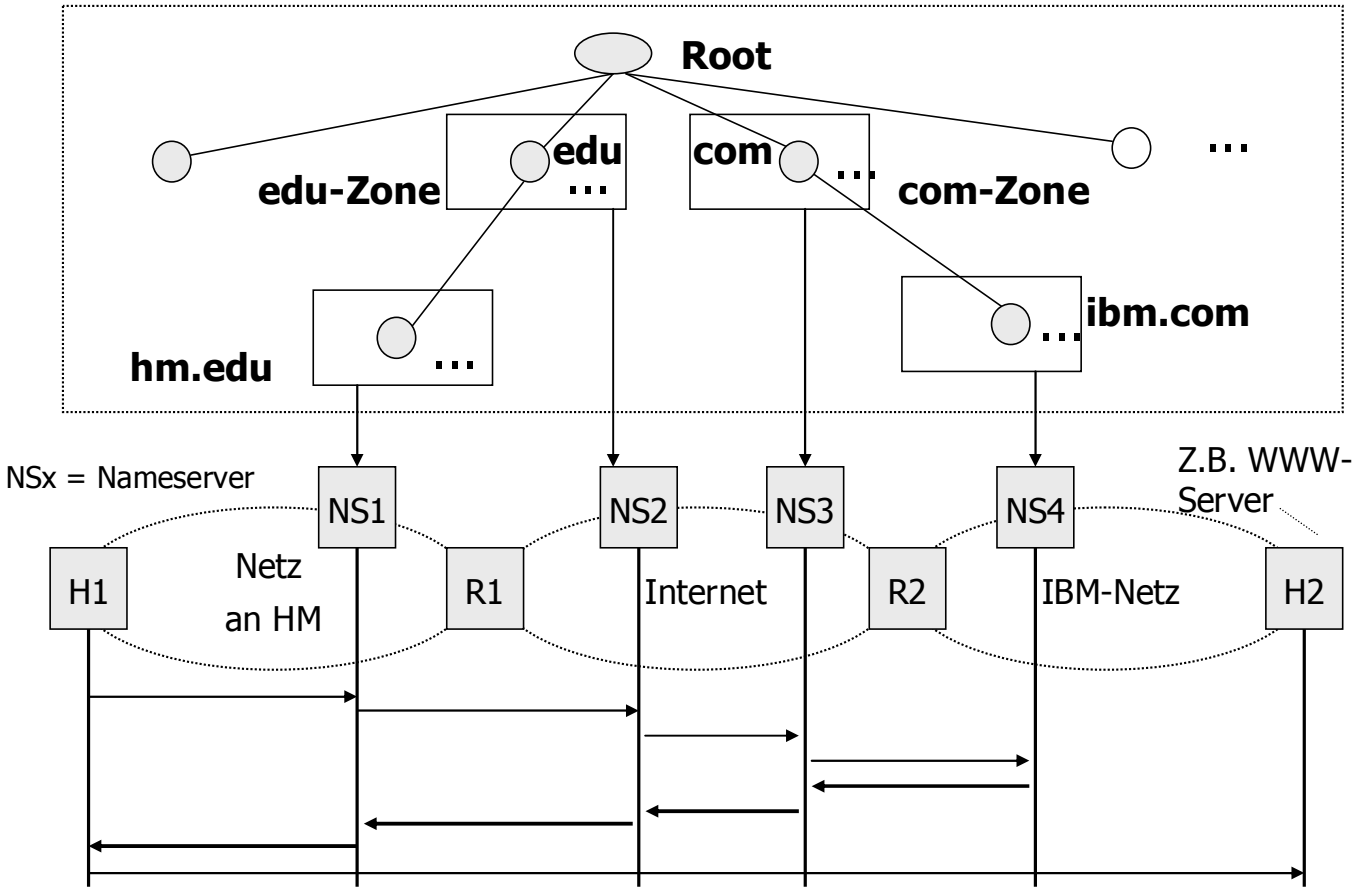
- DNS-Namen bestehen aus zwei Teilen, einem Hostnamen und einem Domainnamen, die zusammen den „**Fully Qualified Domain Name**“ (FQDN) bilden
  - Max. 255 Zeichen, keine Sonderzeichen, Umlaute oder Leerzeichen
  - Max. 63 Zeichen pro Teildomain oder Hostname
  - Beispieladresse: www.isys-software.de (isys-software.de ist der Domainname)
- Zu jeder Domain gehört eine Körperschaft, die diese verwaltet und für die Namensvergabe zuständig ist, oberste Körperschaft ist das NIC

Beispiel für eine iterative Auflösung einer IP-Adresse



- Nameserver verfügen über eigene Resolver, die üblicherweise iterativ arbeiten

Beispiel für eine rekursive Auflösung einer IP-Adresse



- Resolver von Clients arbeiten rekursiv

DNS Records

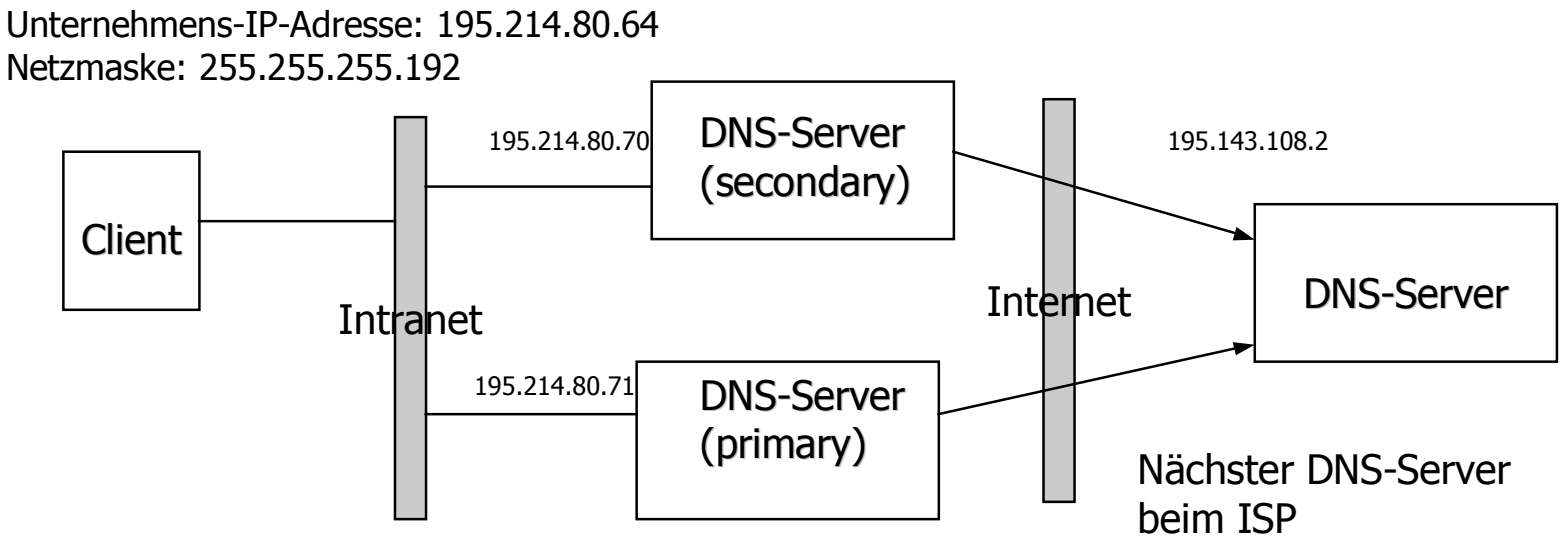
- Aufbau eines Resource Records als Quintupel folgender Form:

(Domainname, Time-to-live, Type, Class, Value)

Domainname:	Name der Domäne
Time-to-live:	Stabilität (Gültigkeitsdauer) des Werts als Integerzahl (je höher desto stabiler), für das Caching wichtig (kurz: TTL)
Type:	Typ des Records (A = IP-Adresse, NS = Name Server Record, MX = Mail-Record, PT= = reverse Record)
Class:	immer IN (Internet)
Value:	Wert des Records je nach Typ: z.B. bei Typ = A → IP-Adresse

Unternehmensnetz anbinden

- Beispiel eines Unternehmensnetzes
- DNS-Server-Eintrag
  - Es gibt zwei DNS-Server mit den Adressen 195.214.80.70 und 195.214.80.71
  - Nächster DNS-Server beim ISP: 195.143.108.2

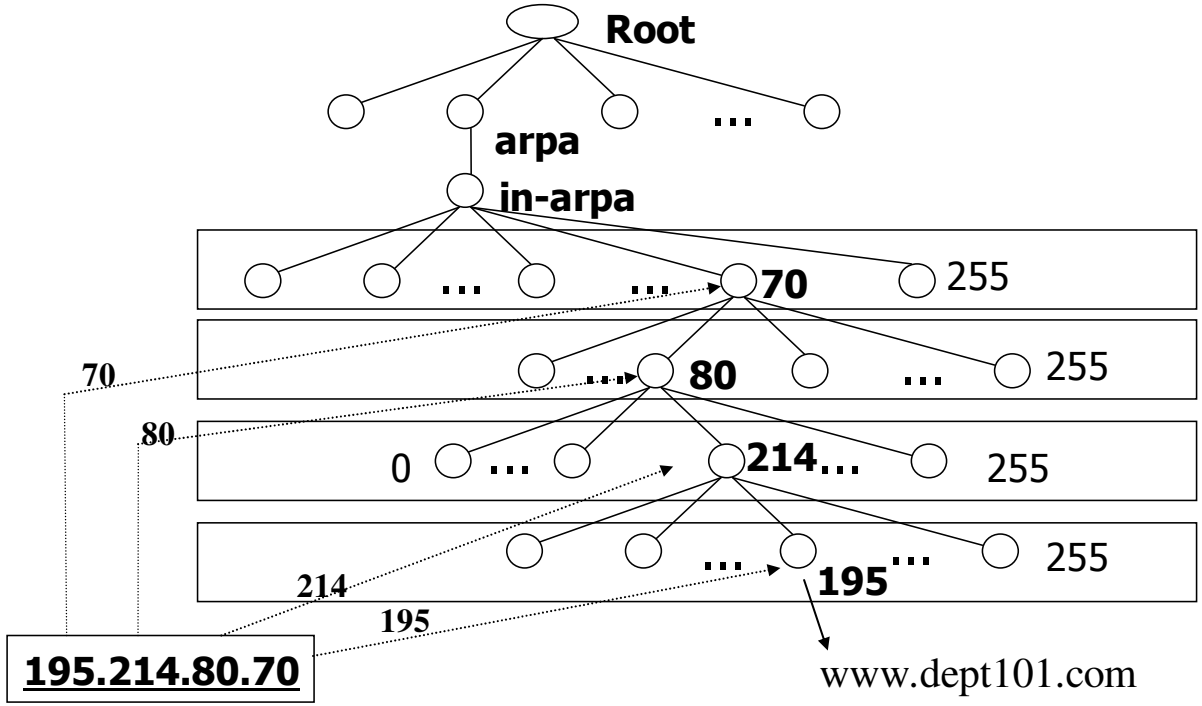


Auflösung von IP-Adressen auf Hostnamen

- Über DNS lassen sich auch IP-Adressen auf Hostnamen auflösen (**inverse Abfrage**)
- Benötigt man z.B. bei der Fehlerbehebung in TCP/IP-Netzen und für Logdateien
- Diese Zuordnungen werden in der besonderen Domäne **in-addr.arpa** durch InterNIC verwaltet
- Die Knoten der Domain sind nach Zahlen in der für IP-Adressen üblichen Repräsentation benannt
  - in-addr.arpa hat 256 Subdomains
  - Die Subdomains haben jeweils wieder 256 subdomains
  - In der untersten (vierten) Stufe werden die Resource Records mit vollem Hostnamen eingetragen

Auflösung von IP-Adressen auf Hostnamen: Reverse DNS

- Aufwändig: Durchsuchen des gesamten Domänen-Baums nach einer IP-Adresse
  - Daher eigene Domäne (Infrastruktur-Domäne): **in-addr.arpa-Domäne**
  - Unterhalb dieser Domäne existieren nur drei Subdomänen-Ebenen



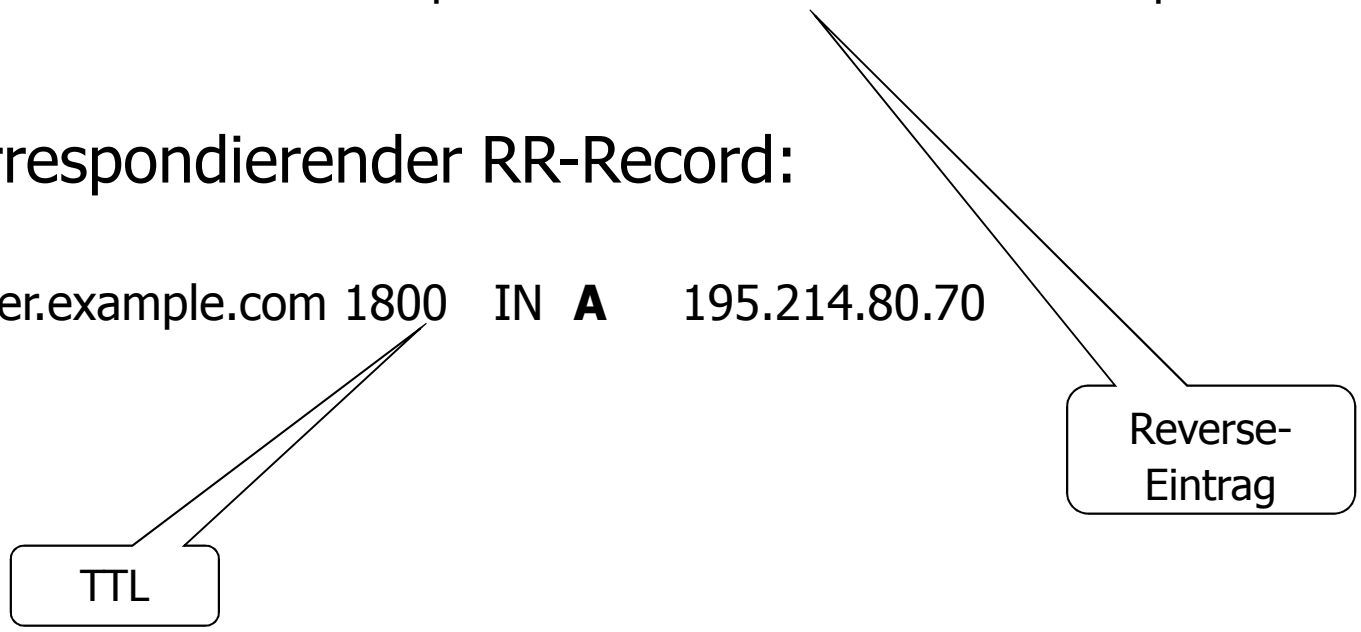
Reverse DNS: RR-Record-Beispiel

- Eintrag für Reverse DNS:

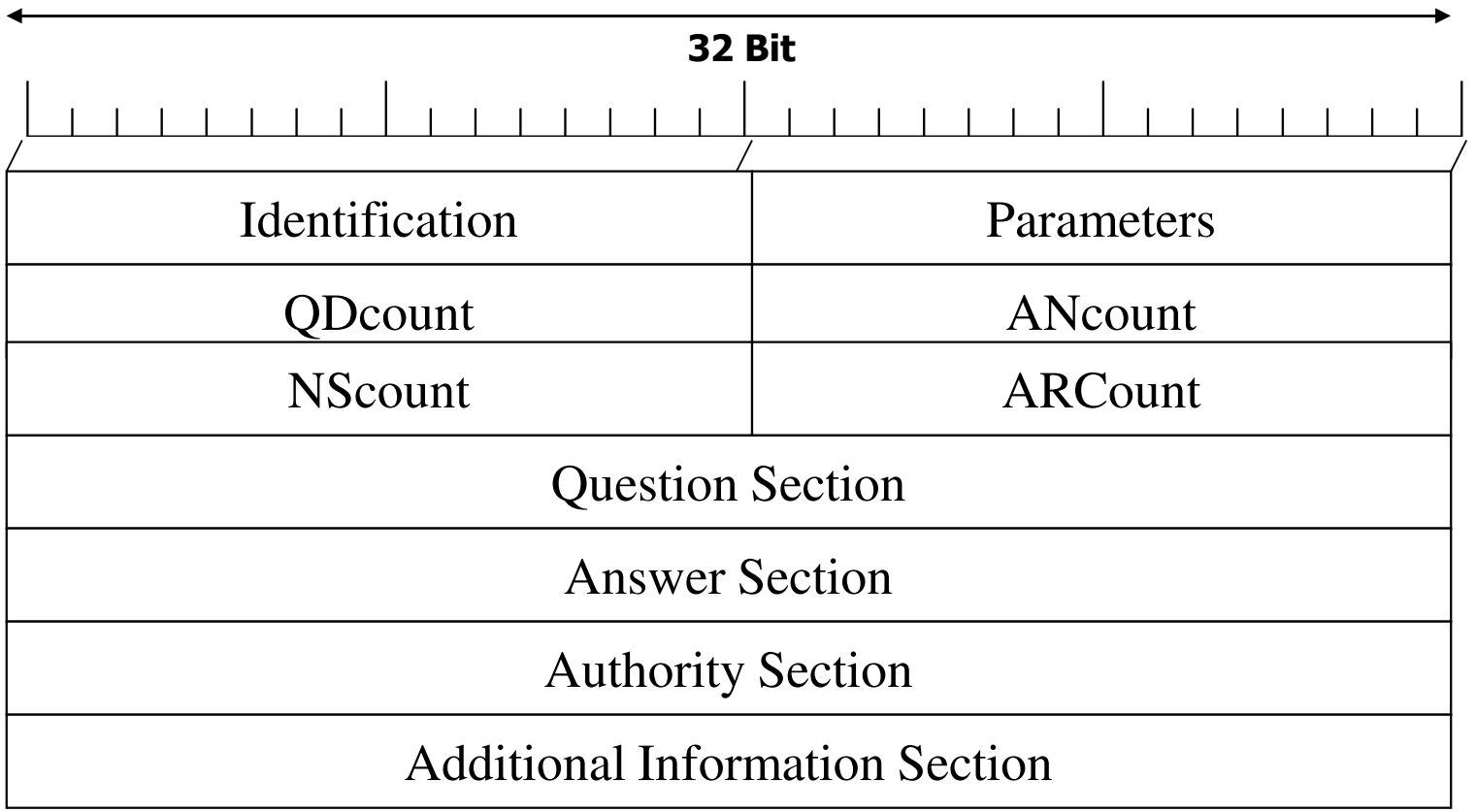
70.80.214.195.in-addr.arpa. 1285 IN **PTR** server.example.com

- Korrespondierender RR-Record:

server.example.com 1800 IN **A** 195.214.80.70



DNS-PDU


32 Bit
Identification
Parameters
QDcount
ANcount
NScount
ARCount
Question Section
Answer Section
Authority Section
Additional Information Section

DNS-PDUs

- DNS-Nachrichten setzen sich aus Sektionen (Sections) zusammen:
  - Der **Header** besteht aus den ersten sechs Feldern (Header Section)
  - **Question Section:** Enhält Felder zur Spezifikation der Anfrage
  - **Answer Section:** Enthält die Antwort eines Name-Servers in Form von Resource Records (RRs)
  - **Authority Section:** Enthält RRs eines autorisierten Servers
  - **Additional Information Section:** Enthält zusätzliche Informationen zur Anfrage oder zur Antwort

DNS-Header

- **Identification (2 Bytes)**
  - Id der Anwendung, die die Abfrage abgesetzt hat
- **Parameters**
  - 0 := Anfrage, 1 := Response
  - Zusätzliche Information: Hinweis, ob es eine normale oder eine inverse Abfrage handelt
  - Inverse Abfrage: Für IP-Adresse einen Hostnamen suchen
- **QDcount**
  - Anzahl der Einträge in der Question Section
- **ANCount**
  - Anzahl an Resource Records (RR) in der Authority Section
- **ARcount**
  - Anzahl an RRs in der Additional Information Section

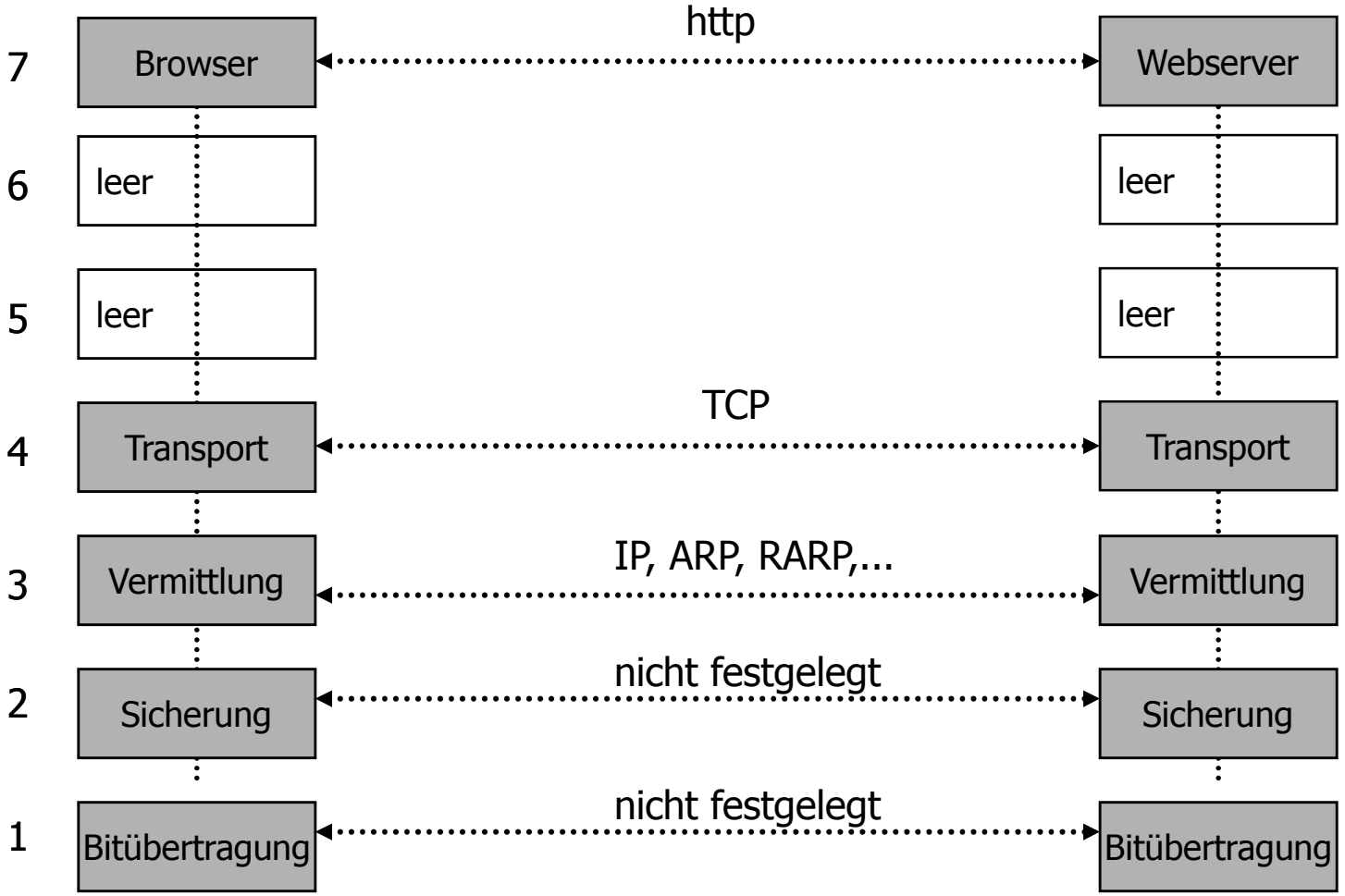
Exkurs:
DNS-Paketlänge unter Windows 2003

- Ändern der UDP-Paketlänge für DNS unter Windows 2003:
  - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services**DNS**\Parameters → **MaximumUdpPacketSize**
  - Standardwert: 1280 Byte
  - Werte zwischen 512 and 16384 sind möglich

Überblick

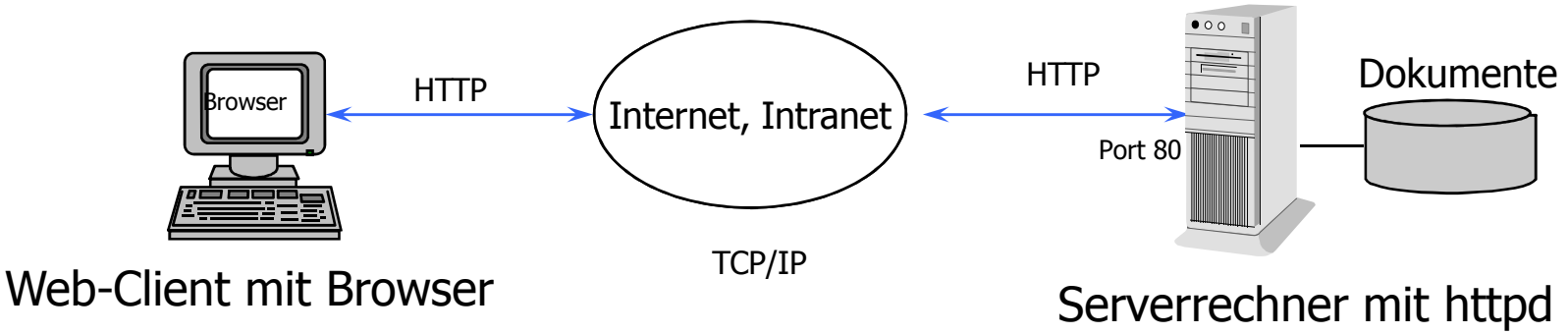
1. Client-/Server versus Peer-to-Peer (P2P)
2. Domain Name System
3. **HTTP**
4. E-Mail
5. Multimedia-Protokolle

Überblick
Einordnung in die TCP/IP-Protokollfamilie



Web-Basiskomponenten

- HTML (Hypertext Markup Language) dient als Auszeichnungssprache (Markup Language)
- Dokumente (auch HTML-Seiten genannt) können über Links beliebig verknüpft werden
- HTTP (Hypertext Transfer Protocol) auf Basis von TCP/IP als textbasiertes Kommunikationsprotokoll
- Web-Browser auf der Clientseite als grafische Benutzeroberfläche
- Web-Server (httpd) auf der Serverseite mit Dokumenten (HTML)



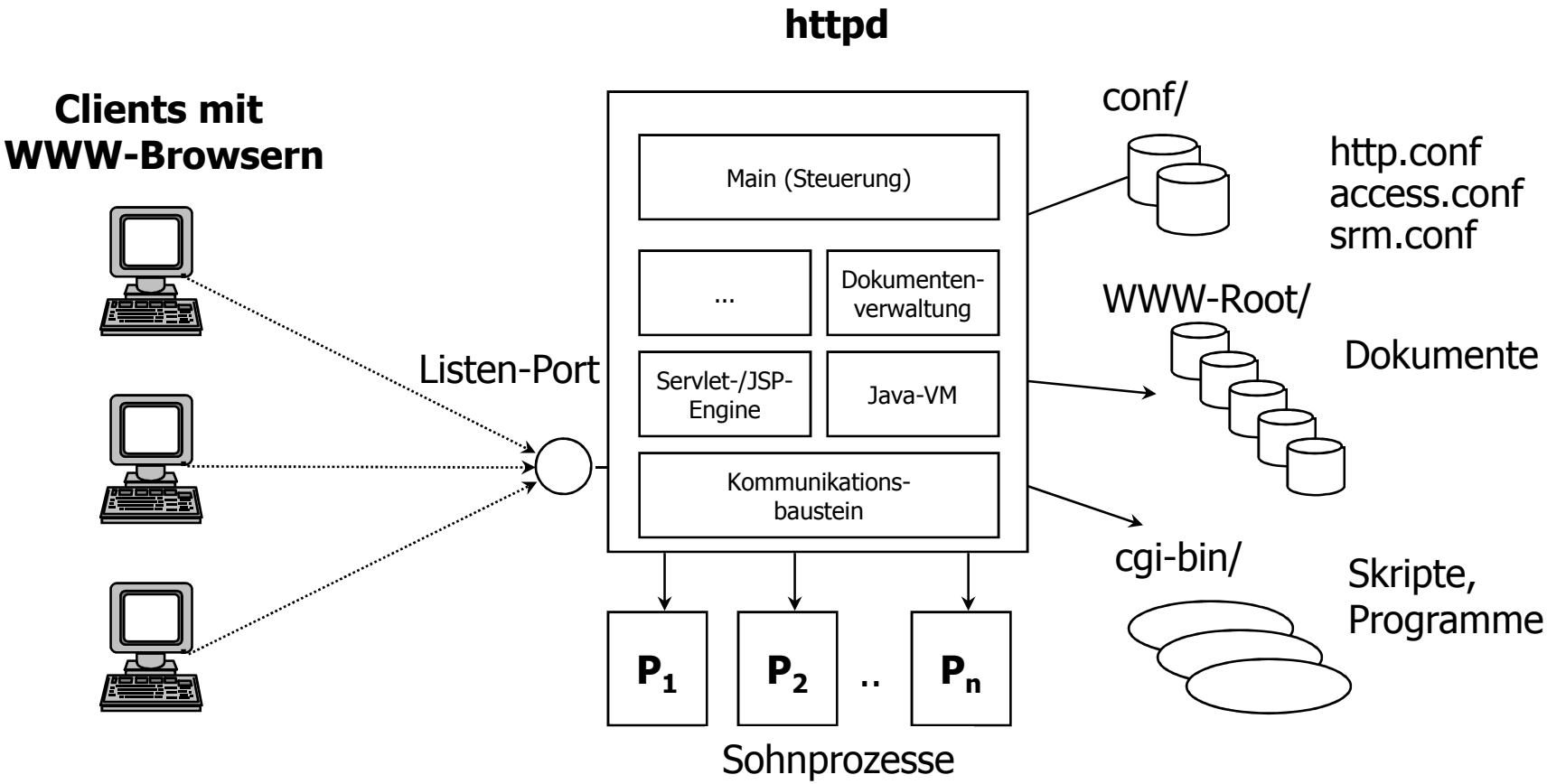
Der Web-Server: Arbeitsweise

- Web-Server wartet auf ankommende Verbindungsaufbauwünsche standardmäßig auf TCP-**Port 80** mit Socket-Funktion listen()
- Verbindungsaufbauwunsch wird entgegengenommen
- HTTP-Request wird empfangen und in einem eigenen Thread oder Prozess bearbeitet
- Das gewünschte Dokument wird lokalisiert (wir nehmen hier die Adressierung einer statischen Webseite an)
- Hierzu wird im konfigurierten Root-Verzeichnis des Web-Servers gesucht

Der Web-Browser: Grundlegendes

- Web-Browser stellen die Clients dar und bieten dem Anwender eine komfortable Benutzeroberfläche
- Die gängigsten Web-Browser sind:
  - Microsoft Internet Explorer
  - Mozilla Firefox
  - Apple Safari
  - ...
- Web-Browser sind komplexe Softwareprogramme mit vielen Möglichkeiten der Einstellung
- Sie können erweitert werden: Plug-ins
- Sie arbeiten nicht immer gleich →schafft Probleme bei der Entwicklung von Web-Anwendungen

Zusammenspiel der Komponenten



Ein einfaches HTML-Beispiel

```
<HTML>
<HEAD>
    <TITLE> Beispiel einer Web-Seite </TITLE>
</HEAD>
<BODY>
<H1>  Test-Webseite </H1>
<H2> <I> Interessante Links: </I></H2>
<UL compact>
<LI><A HREF="http://www.isys-software.de"> iSYS Web-Site</a> </li>
<LI><A HREF="http://www.fh-muenchen.de"> Web-Site der FH-M&uuml; nchen</a>
    </li>
</UL>
</BODY>
</HTML>
```

Kompakt darzustellende Liste

Listeneintrag

Hyperlink

- Textbasierte Auszeichnungssprache (Markup Language)
- HTML Version 5

HTTP, Einführung

- HTTP (= HyperText Transfer Protocol) bildet als Kommunikationsprotokoll zwischen Web-Client- und Server das **Rückgrat** des Webs
- Man muss HTTP verstehen, wenn man gute Web-Anwendungen entwickeln will
- RFC 2616 (vorher 2068) der Network Working Group regelt den Protokollstandard
- HTTP ist ein **verbindungsorientiertes** Protokoll
- Nutzung von **MIME**-Bezeichnern für content-type (Multipurpose Internet Mail Extensions), z.B. text/html, image/gif

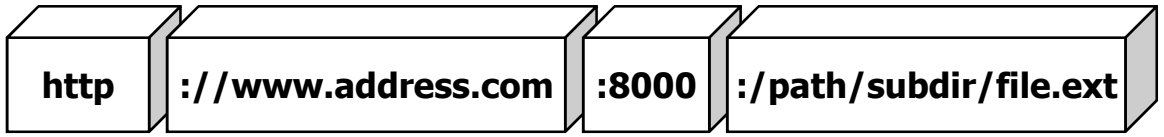
Das Protokoll, Einführung

- Für jede Seite und jede Grafik, die vom Server gelesen wird, wird in HTTP V1.0 standardmäßig eine TCP-Verbindung zwischen WWW-Browser und WWW-Server aufgebaut werden
  - Nicht mehr so in HTTP V1.1 (1999)
- HTTP ist vom Protokolltyp her ein **zustandsloses Request/Response-Protokoll**
  - Weder Sender noch Empfänger merken sich irgendwelche Stati zur Kommunikation
  - Ein Request ist mit einem Response vollständig abgearbeitet
- **Protokolloperationen** sind z.B. GET, HEAD, PUT, POST,...

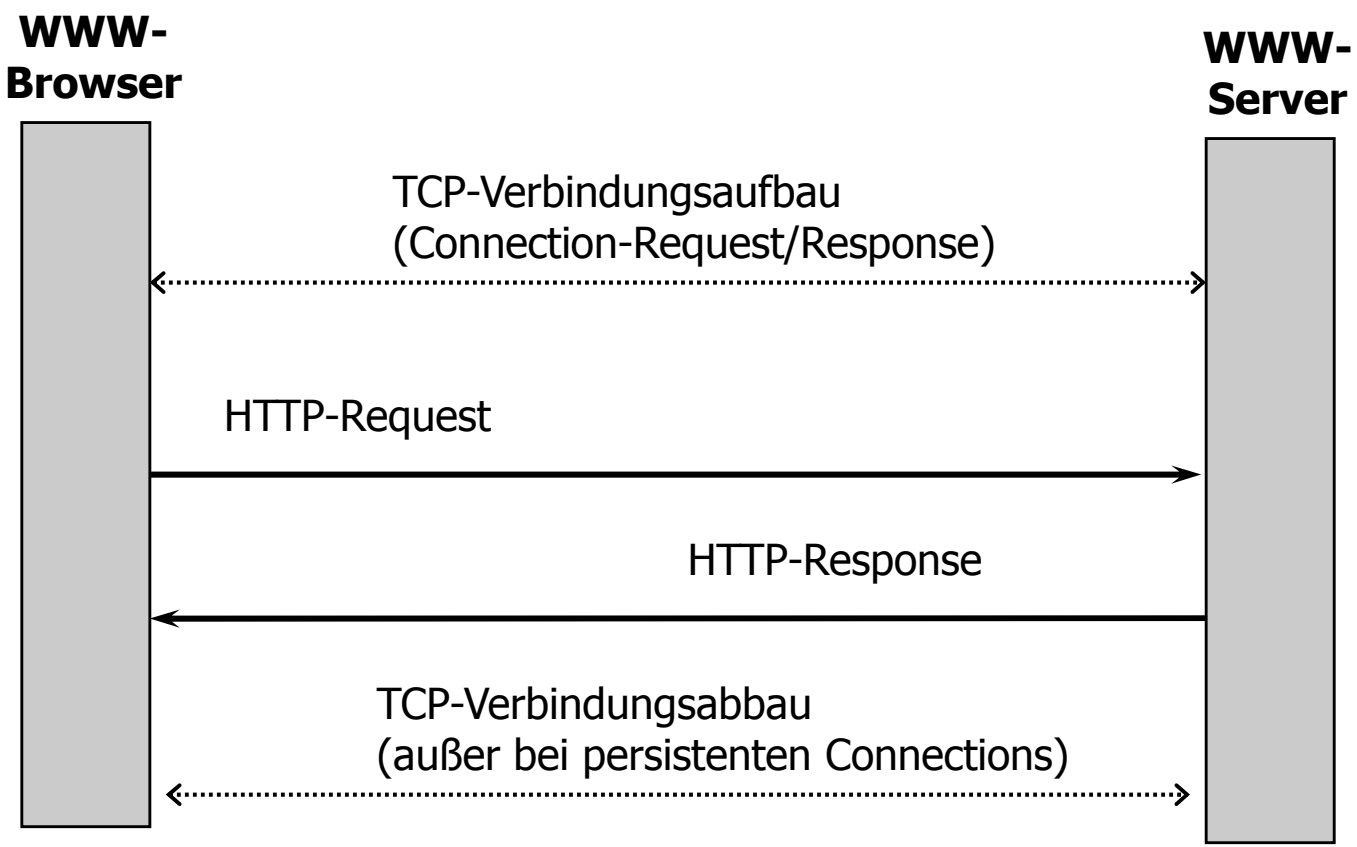
Adressierung im WWW

- Die Adressierung über URIs bzw. URLs
- Uniform Resource Locator (URL) identifiziert das Dokument
- Uniform Resource Identifier (URI): Allgemeinerer Begriff als Überbegriff für alle Adressierungsmuster, die im WWW unterstützt werden

URL-Aufbau:

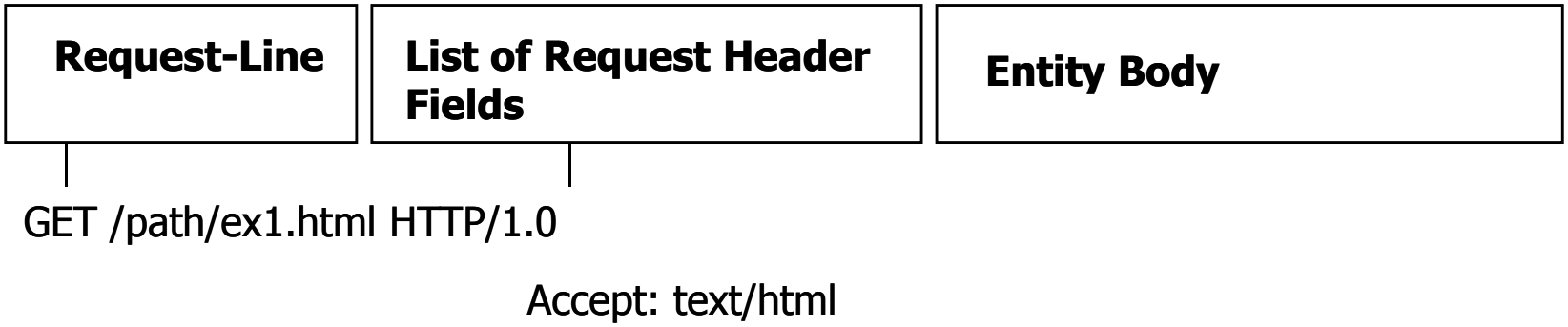


Ablauf der Kommunikation

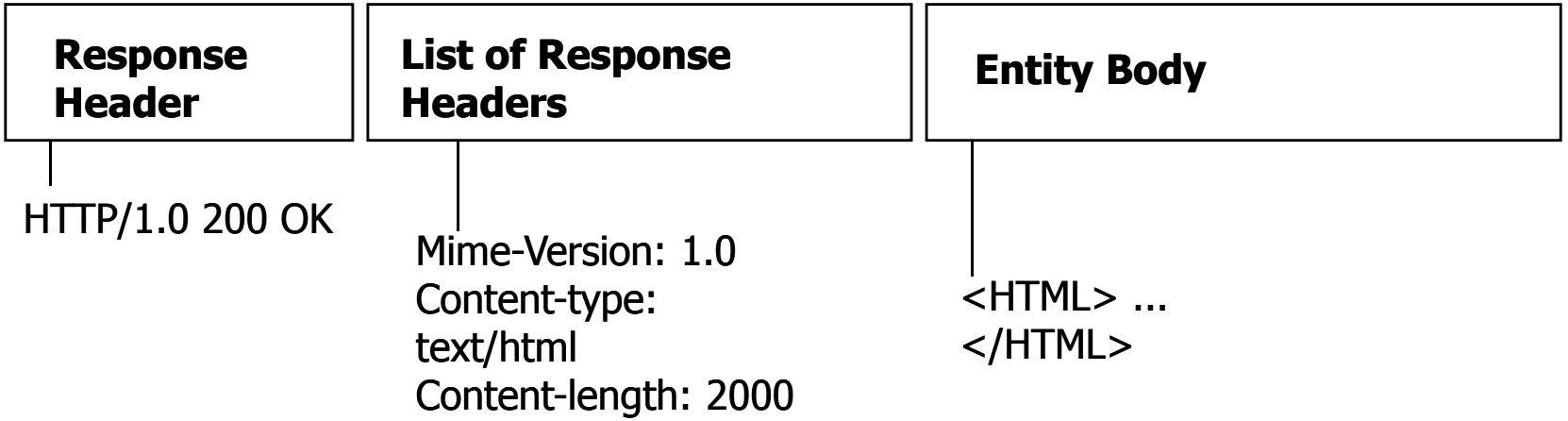


Aufbau der HTTP-PDUs

HTTP-Request



HTTP-Response



Ablauf der Kommunikation

Vom Client initiierte Aktionen:

1. Browser ermittelt Hostnamen des Servers aus URI und besorgt sich über DNS die IP-Adresse des Servers
2. Client baut aktiv eine TCP-Transportverbindung zum Socket des Servers mit Portnr. 80 auf.
3. httpd im Server nimmt die Verbindung an.
4. Client sendet HTTP-Befehl, z.B. GET /index/html HTTP/1.1
5. Client sendet weitere optionale HTTP-Header (eigene Konfiguration und akzeptierte Dokument-Formate,..), z.B. Accept: image/gif
6. Client sendet eine Leerzeile (Ende des Requests anzeigen)
7. Client sendet evtl. noch zusätzliche Daten als Parameter (Input-Felder aus Forms)

Ablauf der Kommunikation

Vom Server initiierte **Aktionen**:

1. Nach der Bearbeitung des Requests sendet der Server eine Statuszeile (Response Header) mit drei Feldern (Version, Statuscode, lesbarer Text), z.B. HTTP/1.1 200 OK
2. Anschließend sendet der Server HTTP-Headers wie z.B.
   Content-type: text/html
   Content-length: 2482
3. Danach wird eine Leerzeile gesendet und das angeforderte Dokument
4. Falls nicht vom Client ein Header „Connection: Keep alive" gesendet wurde, wird die TCP-Verbindung wieder abgebaut

→ in HTTP 1.1 wird „Keep alive" standardmäßig verwendet. Dadurch können können Applets, Graphiken, Rahmen,... über dieselbe Verbindung übertragen werden

Übung

Da HTTP ein ASCII-basiertes Protokoll ist, kann man den Ablauf der Kommunikation auch mit einer Telnet-Verbindung verfolgen. Probieren Sie dies an einem beliebigen WWW-Server aus.

- telnet <url des Servers> 80
- Was passiert?
- Fordern Sie eine WWW-Seite mit GET-Operation an
- Was passiert?

Protokolloperation GET und HEAD

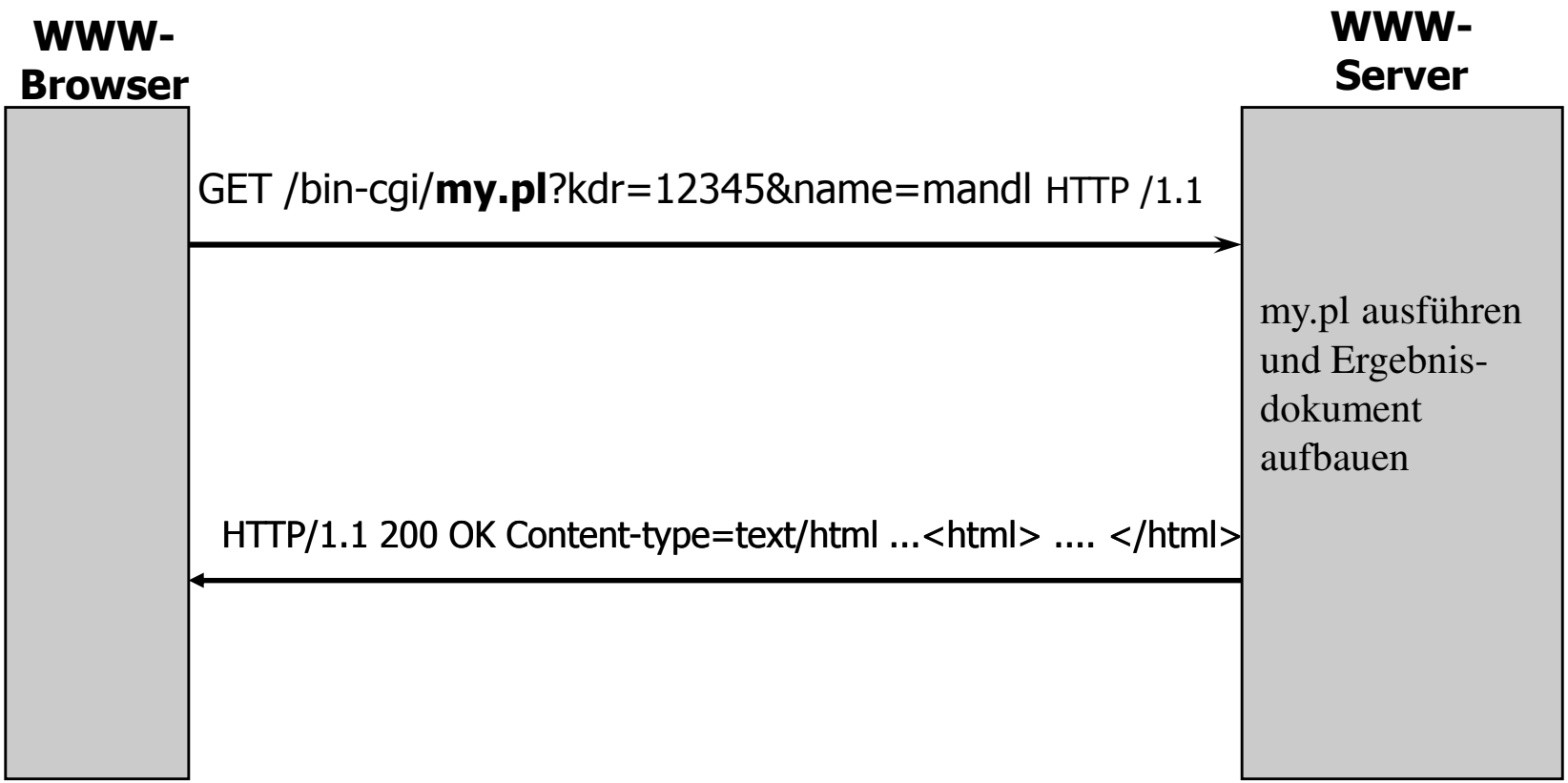
- Wichtig sind vor allem die Operationen GET und POST
- Bei GET ist der Body-Bereich immer leer und zur Übertragung von Input-Parametern für serverseitige Programme wird die URL ergänzt (nach einem Fragezeichen kommen die Parameter)

  Beispiel: GET /bin-cgi/my.pl?kdr=12345&name=mandl HTTP 1/1
- Im Beispiel sendet der Server nach Abarbeitung des CGI-Perlscripts my.pl den Output des Scripts als Response zurück.
- Die HEAD-Operation funktioniert wie GET nur dass hier vom Server kein Ergebnisdokument gesendet wird

  Beispiel: HEAD /index/html HTTP/1.1

  → Sinnvoller Einsatz z.B. dann, wenn Client nur das Datum der letzten Änderung oder die Dokumentgröße wissen will

Beispiel einer GET-Operation



Protokolloperation POST

- Über die POST-Methode können in einem Client-Request Daten an den Server zur Weiterverarbeitung über ein Programm gesendet werden
- Die Daten werden im Body gesendet
- Der Server gibt die Daten an das mit der angegebenen URI adressierte Programm weiter. Meist werden die Daten URL-codiert übergeben
- Am Response ändert sich nichts

Beispiel:
```
POST /cgi-bin/my2.pl HTTP/1.1
Content-type: application/x-www-form-urlencoded
Content-length: 20
monat=oktober&tag=24
```

Header

- Es gibt verschiedene Header-Kategorien, die eine Menge von Möglichkeiten bieten. Header-Kategorien sind:
  - Allgemeine Header für Client und Server
  - Request Header für Client
  - Response Header für Server
  - Entity Header für Client und Server
- Aufbau von Headern (Groß-/Kleinschreibung spielt keine Rolle):

  <header-name>: <header-value>

  z.B. Content-Type: text/html oder content-type: text/html
- Header kann sich über mehrere Zeilen erstrecken, Folgezeilen müssen mit Blank oder Tab beginnen

Header: Ausgewählte Beispiele

- Allgemeine Header
  - Cache-Control: Direktiven → Definiert in einer Liste Caching-Direktiven (Liste mit Komma abgetrennt zulässig)

    z.B. im Request-Header: Cache-Control: no-cache → keinen Cache verwenden

    z.B. im Response-Header: Cache-Control: no-cache → nicht in den Cache legen
- Request-Header für Client
  - Accept: <typ>/<untertyp> (Liste mit Komma abgetrennt zulässig)

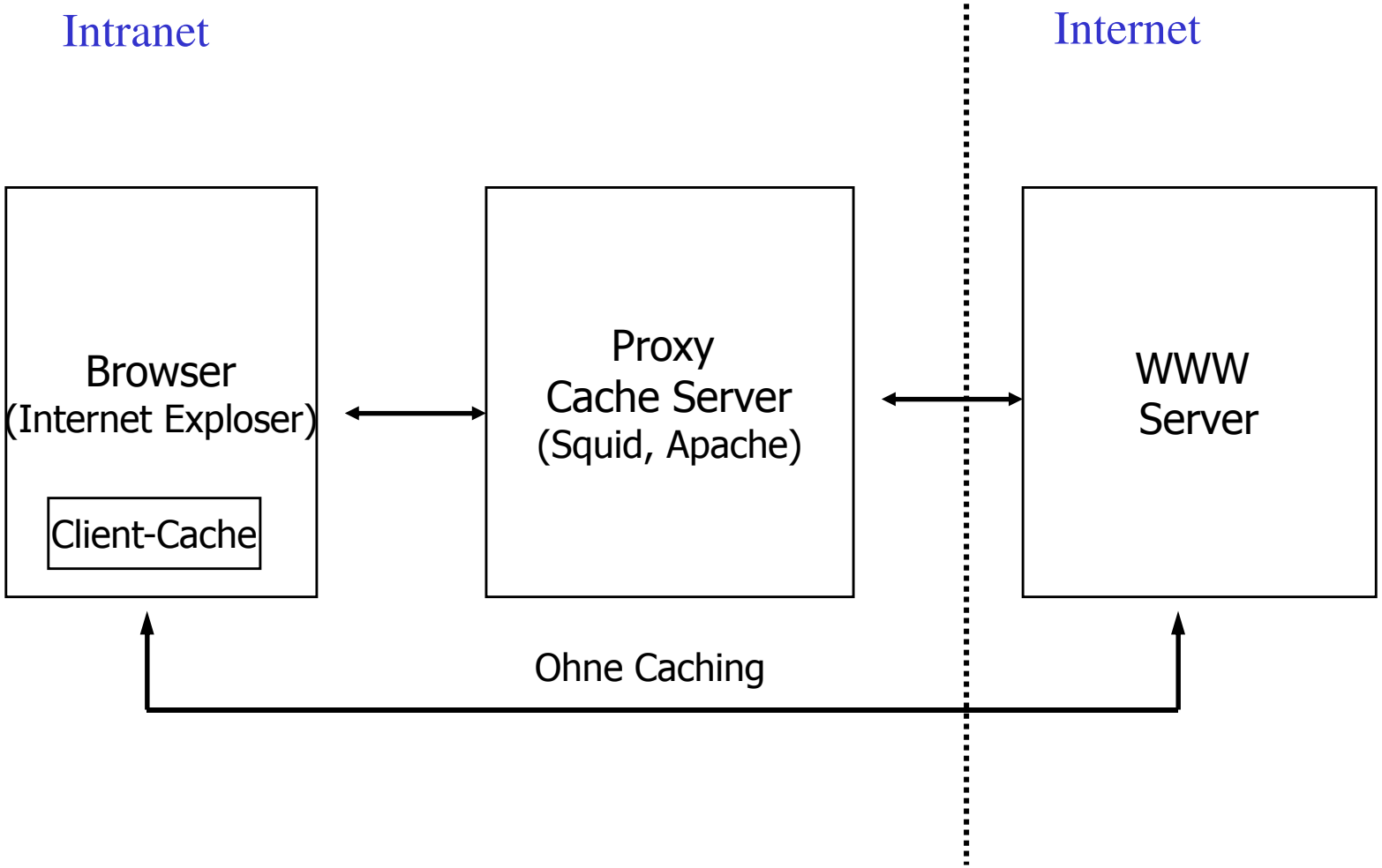
    z.B. Accept: text/* → Alle Textuntertypen sind erlaubt (html, plain, enriched,...)
  - Cookie: <name>=<wert> (Liste mit Strichpunkte abgetrennt ist zulässig). Speichert das Name-Value-Paar für die URL

    z.B. Cookie: mps=1234123

Proxy Caching

- Z.B. Squid oder Apache Cache
- Arbeitsweise ist im HTTP-RFC definiert
- Gründe für Caching:
  - Performance-Optimierung
  - Sicherheitsaspekte bei Firewalleinsatz
  - Begrenzte Anzahl an vorhandenen IP-Adressen
- In Mehrbenutzerumgebungen kann ein „gecachtes" Dokument mehreren Web-Clients zur Verfügung gestellt werden: Erhöhung der Hitrate
- Regeln für das Caching sind konfigurierbar

Proxy Caching

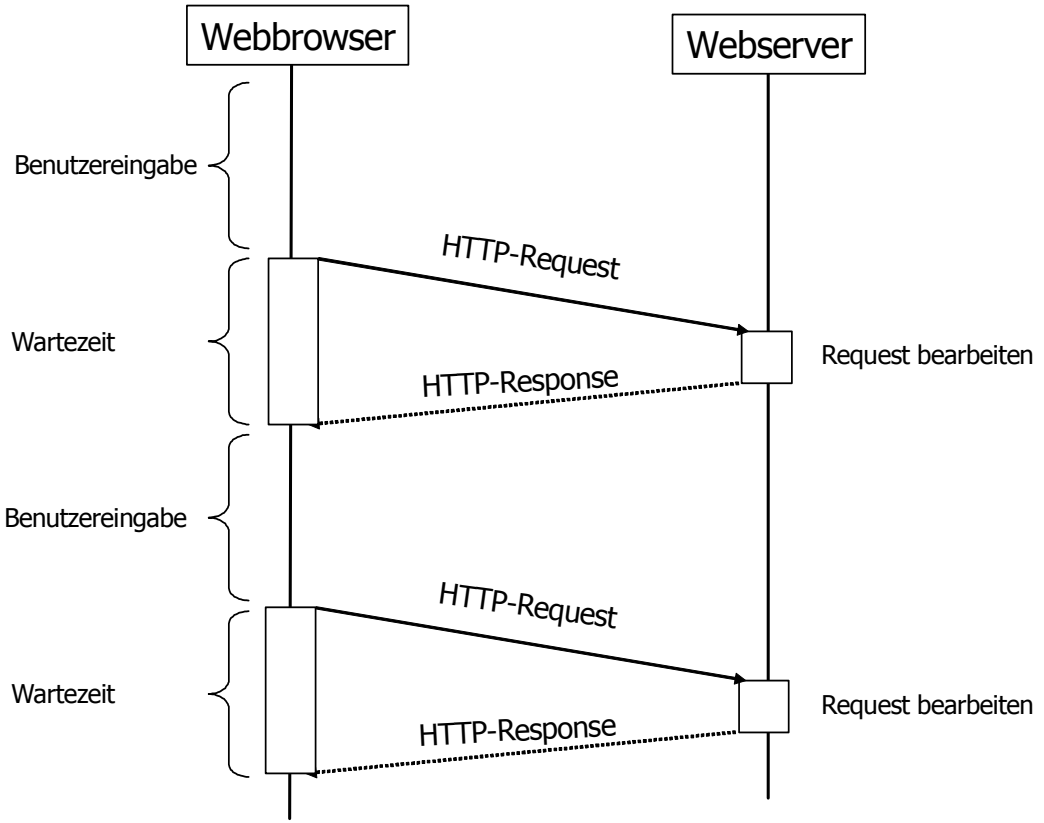


AJAX - Einführung

- Asynchronous JavaScript and XML
- Der Begriff geht aus einem Artikel von Jesse James Garrett hervor „Ajax: A New Approach to Web Applications“
- Es beschreibt keine neue Technologie, sondern einen neuen Ansatz, der auf bereits vorhandenen Technologien aufbaut
- Die traditionelle Webkommunikation soll durch asynchrone Mechanismen benutzerfreundlicher gestaltet werden

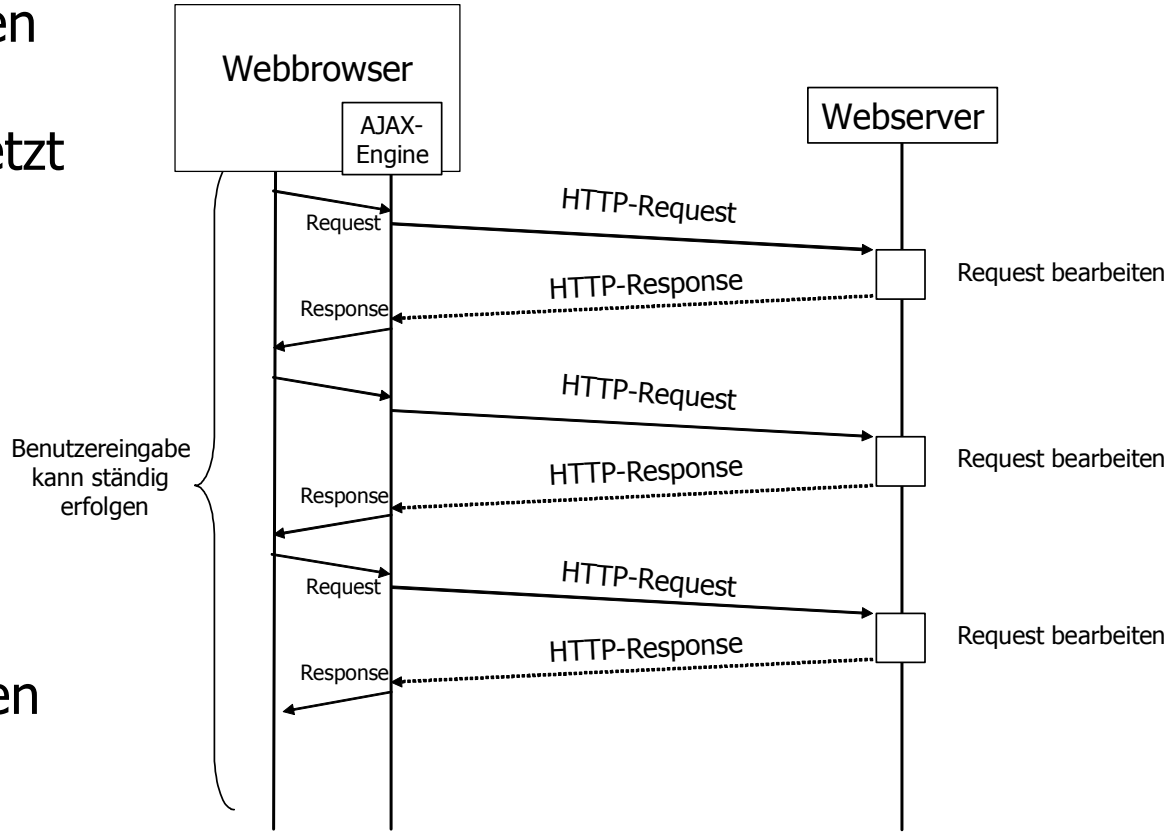
Herkömmliche Webkommunikation

- Formularbasiert
- Es wird immer die komplette Seite übertragen
- Nach jeder Anfrage muss der Benutzer warten und kann nicht interagieren

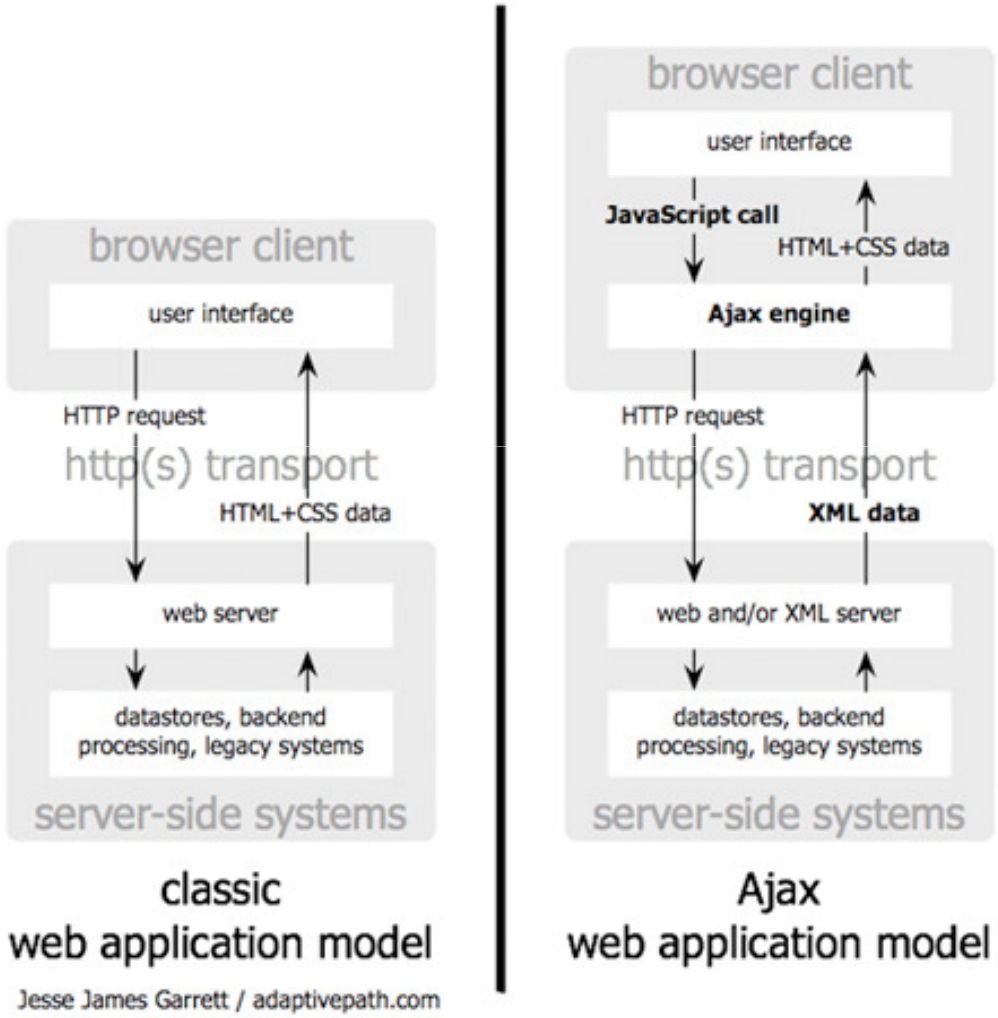


AJAX – asynchrone Webkommunikation

- Die Anfragen werden asynchron im Hintergrund abgesetzt
- Es werden nur Nutzdaten versandt
- Die Antworten werden in die Seite eingearbeitet
- Der Benutzer kann jederzeit interagieren



AJAX – Kommunikationsablauf



AJAX – XMLHttpRequest

- Zentraler Baustein von AJAX
- Ermöglicht die asynchrone Kommunikation
  - Absenden von http-Requests
  - Überwachen der Kommunikation und der eingehenden Antwort über Event-Handler
- Integration im IE über ActiveX, bei Mozilla nativ im JavaScript Dialekt
- Der Zugriff erfolgt JavaScript

AJAX – XMLHttpRequest – Methoden und Attribute

Name	**Beschreibung**
readyState	Der Status des Request-Objekts
onreadystatechange	Dieser Event-Handler wird aufgerufen, wenn sich der Status des Request-Objekts (readyState) ändert.
responseText	Die Nutzdaten der Antwort als Text
responseXML	Die Daten als DOM-Objekt (Typ document). Ist nur gefüllt, wenn der Server die Daten als XML sendet.
status	der HTTP-Status-Code der Anfrage[1]
open(method, url, async, user, pwd)	Bereitet das Request-Objekt auf eine Übertragung an die Adresse URL mit der HTTP-Protokolloperation(GET/POST) vor. Zusätzlich wird angegeben ob die Anfrage asynchron oder synchron gesendet wird(true/false). Optional werden hier User/Passwort zur Authentifizierung über HTTP angegeben.
abort()	Bricht den Request ab.
send(requestBody)	Sendet den über open() initialisierten Request mit dem übergebenen responseBody(nur bei POST) ab.
setRequestHeader(name,wert)	Setzt ein Request-Header-Attribut

AJAX – XMLHttpRequest – readyState

Wert	Bezeichnung	Beschreibung
0	UNSENT	Status nach der Instanzierung des Objekts
1	OPENED	Das Objekt ist bereit die Anfrage zu senden.
2	HEADERS_RECEIVED	Die Anfrage wurde gesendet.
3	LOADING	Header und Status sind verfügbar. Der Message-Body wird empfangen.
4	DONE	Die Anfrage ist beendet. Alle Daten liegen nun vor und können abgerufen werden

AJAX – XMLHttpRequest – Ein einfaches Beispiel

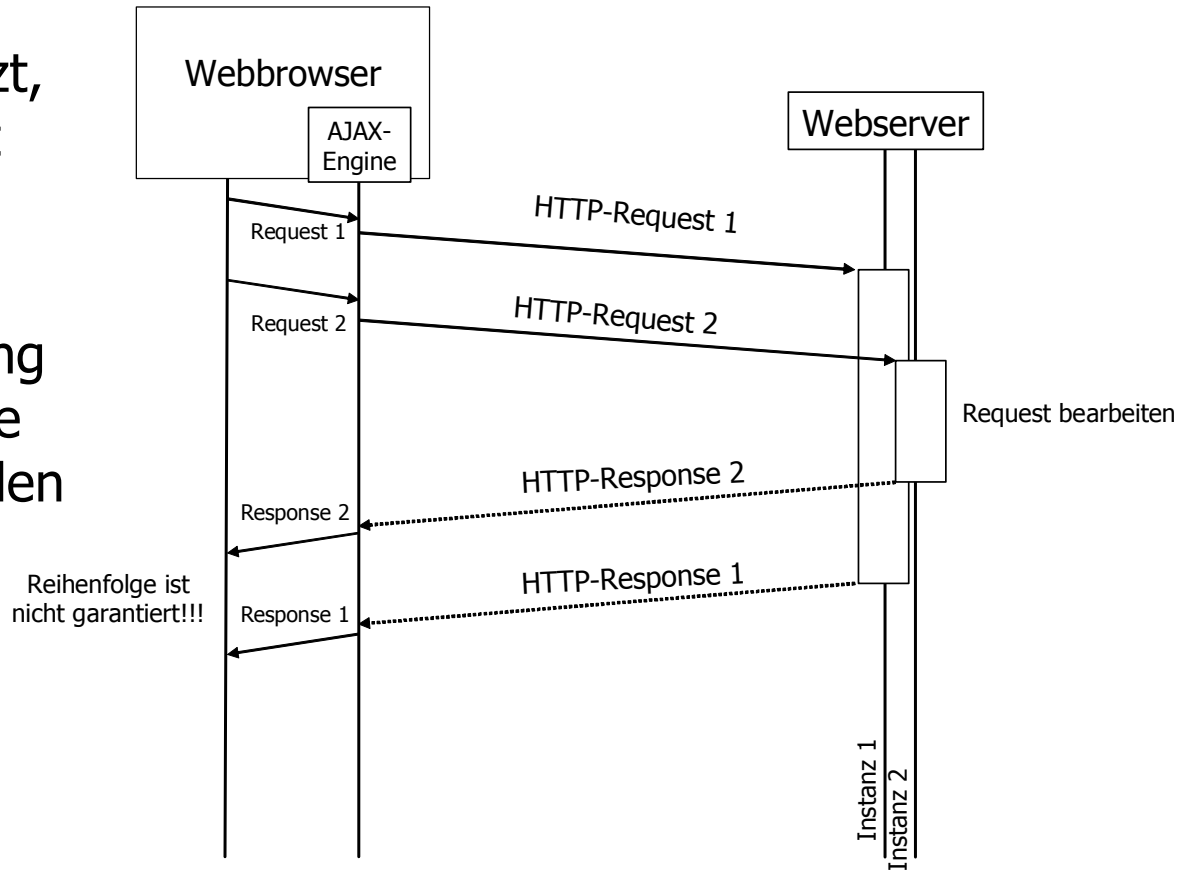
- Beim Drücken des Buttons wird JavaScript-Funktion „loadZitat()“ aufgerufen und die Anfrage an den Server gesendet
- Die Antwort wird nach ca. 3 Sekunden empfangen und auf der Seite angezeigt
- In der Zwischenzeit kann der Benutzer weiter auf der Seite arbeiten, ist als nicht blockiert

AJAX – XMLHttpRequest – Eventhandler

```
function loadZitat(){
var request = new XMLHttpRequest();
request.onreadystatechange= function(){
 if(request.readyState==1){
   document.getElementById("zitat").innerHTML=
         "Warte auf Antwort...";
 }
 if(request.readyState==4){
    if(request.status==200){
     document.getElementById("zitat").innerHTML=
         "Zitat des Tages: \n" +request.responseText;
     }
 }
}
request.open("GET","zitat.php", true);
request.send(null);
}
```

AJAX – Konkurrierende Requests

- Request 1 wird vor Request 2 abgesetzt, die Antwort kommt jedoch später an
- Die Synchronisierung muss auf Clientseite implementiert werden



AJAX – Vor-/Nachteile

Vorteile	**Nachteile**
Es werden weniger Daten, nur Nutzdaten, ausgetauscht.	Viele kleine Anfragen, zusätzliche Netzlast
Daten werden im Hintergrund ausgetauscht, Anwendung ist nicht blockiert	Kompliziertes State-Management, Keine Reihenfolgegarantie.
Einzelne Validierungen von Formularfeldern schon während der Eingabe (Benutzerfreundlichkeit)	Applikationslogik muss im Browser implementiert werden (JavaScript)
Benutzerfreundliche UI-Elemente	Browser-Inkompabilitäten

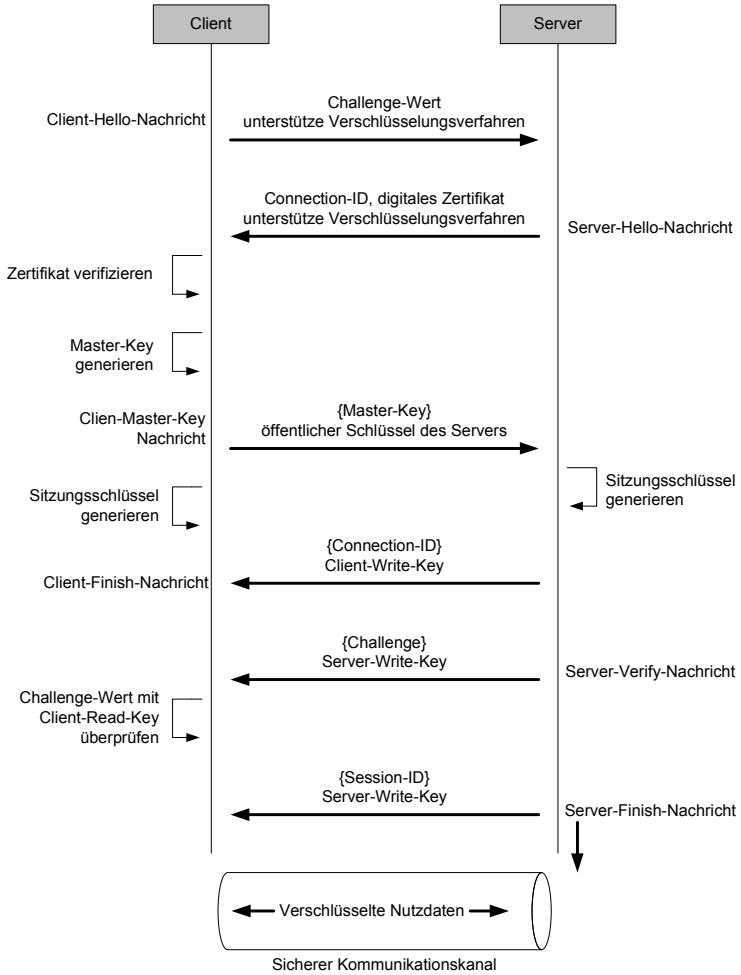
AJAX – Frameworks

- Zahlreiche Frameworks erleichtert die Entwicklung von benutzerfreundlichen Anwendungen
  - Clientseitig: vielfältige innovative und benutzerfreundliche UI-Elemente werden bereitgestell, z.B. DOJO, Scriptaculous,…
  - Serverseitig: meist in Webframeworks integriert, z.B. ASP.NET AJAX, JSF, Google Web Toolkit
- Bei den meisten Frameworks wird angestrebt, komplett auf JavaScript-Programmierung zu verzichten und Elemente mit browserübergreifender Kompatilität bereitzustellen

Einschub: TLS, SSL

- ***Transport Layer Security*** (TLS) bzw. dessen Vorgängerversion ***Secure Sockets Layer*** (SSL) sind Verschlüsselungsprotokolle
- TLS ist die **standardisierte Weiterentwicklung** der SSL-Version 3.0
- Heute wird SSL/TLS von allen gängigen Web-Browsern und Web-Servern unterstützt
- SSL wurde ursprünglich von der Firma Netscape entwickelt und **1996** an die IETF (Internet Engineering Task Force) übergeben
- **1999** wurde TLS in einer ersten Version im **RFC 2246** verabschiedet wurde

Einschub:
TLS, SSL: Protokollverlauf



Überblick

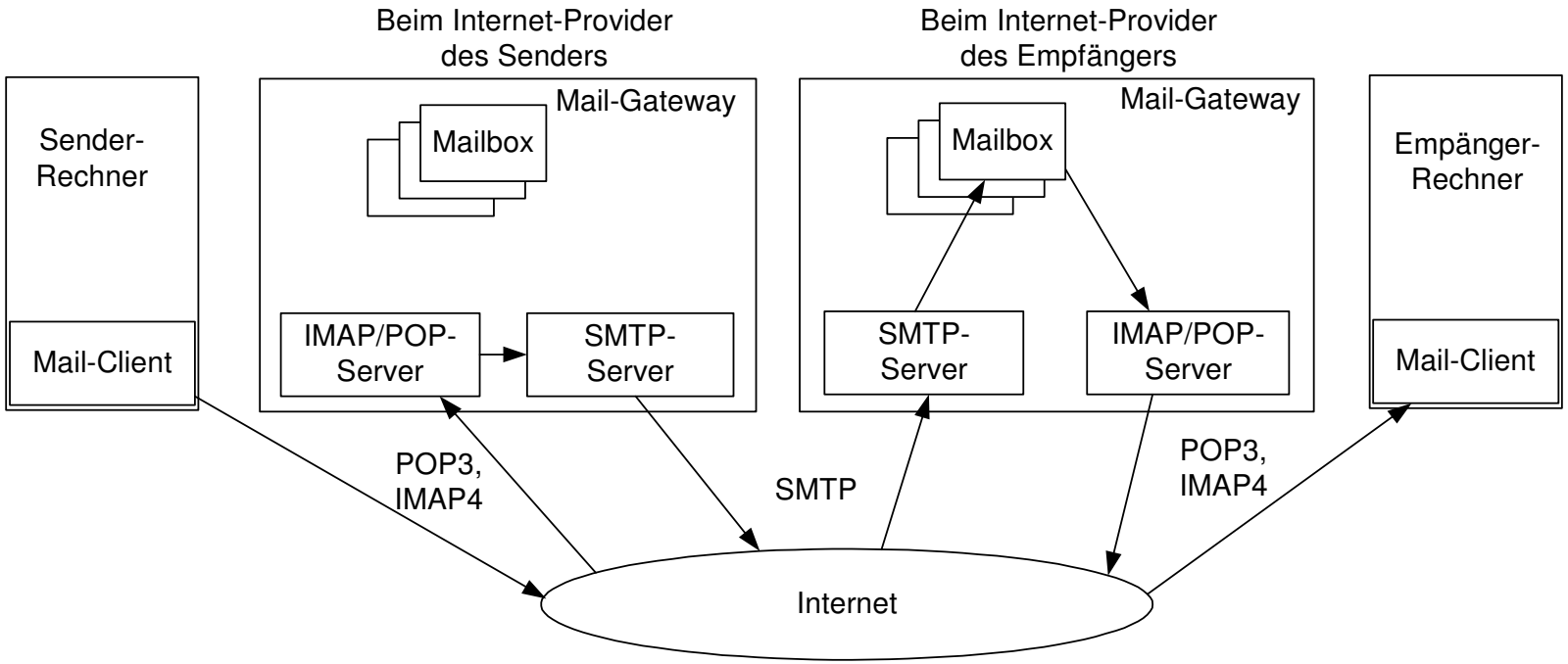
1. Client-/Server versus Peer-to-Peer (P2P)
2. Domain Name System
3. HTTP
4. **E-Mail**
5. Multimedia-Protokolle

Grundlegendes

- Austausch von elektronischen Nachrichten
- Elektronische Mailbox mit einer eindeutigen E-Mail-Adresse (z.B. mandl@cs.hm.edu) in einem **E-Mail-Gateway**
- Die Mail-Gateways kommunizieren untereinander über das Protokoll **SMTP** (Simple Mail Transfer Protocol, well-known TCP-Port = 25)
- **Mailzugangsprotokolle** (Zugang zur Mailbox)
  - **POP3** (Post Office Protocol, Version 3, well-known TCP-Port = 110) → Privatleute
  - **IMAP4** (Internet Message Access Protocol, well-known TCP-Port = 143) → Unternehmen, IMAP kann mehr

Einschub: Architekturüberblick

- Jeder Mail-Client (*Microsoft Outlook*, *Mozilla Thunderbird,...*) baut eine TCP-Verbindung zu seinem SMTP-Server (*sendmail*, *qmail*) bei seinem Internet-Provider auf
- SMTP-Server kommunizieren untereinander



Überblick

1. Client-/Server versus Peer-to-Peer (P2P)
2. Domain Name System
3. HTTP
4. E-Mail
5. **Multimedia-Protokolle**

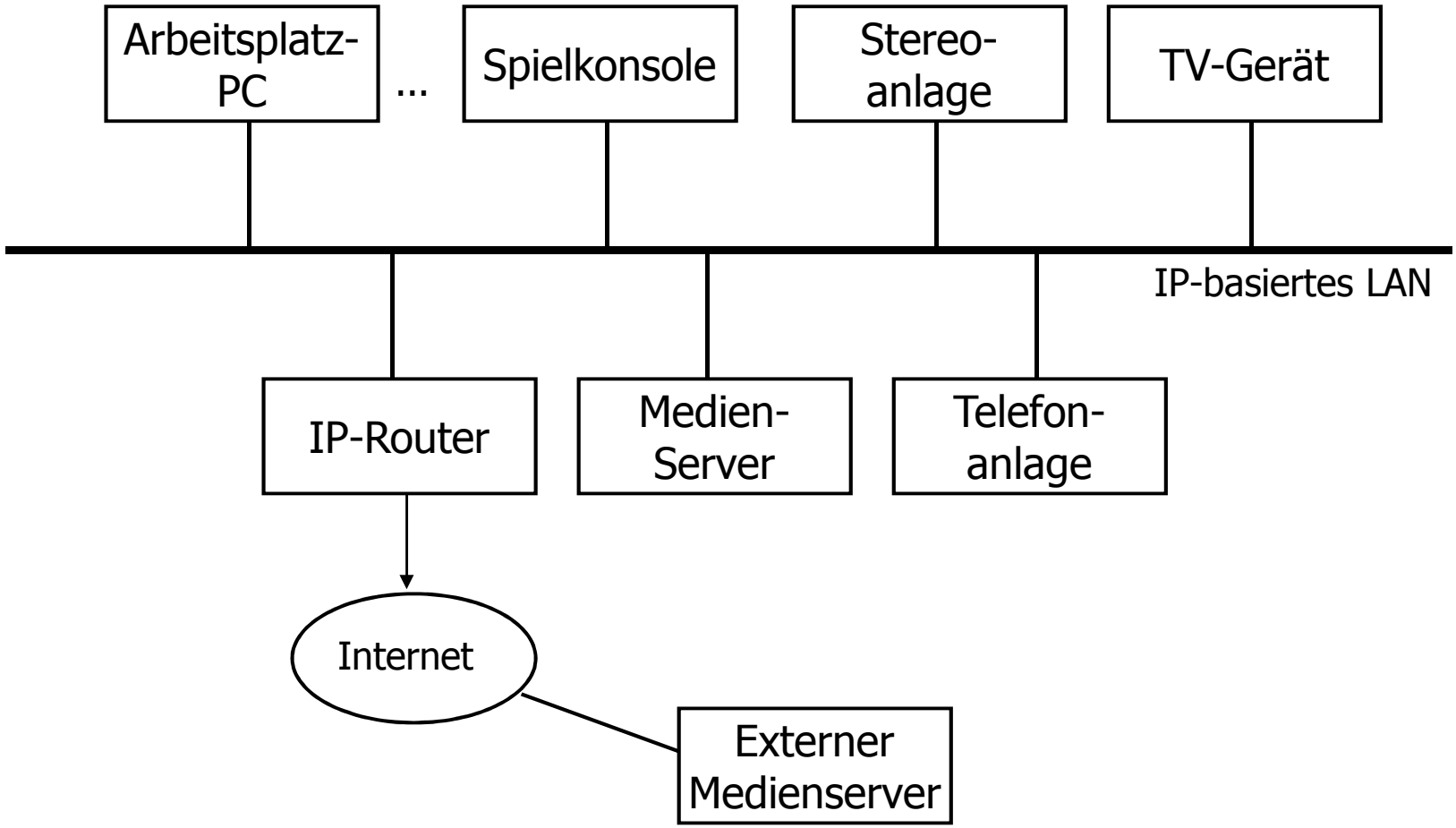
Grundlegendes

- **Multimedia**: Digitale Informationen, die sich aus verschiedenen Medientypen zusammensetze
  - Textdaten, Audio- und Videodaten, Bilder, Fotos usw.
- Multimedia-Verarbeitung hat immense Anforderungen an Netzwerkprotokolle
- **Multimedia-Anwendungen**
  - Video-Übertragungen, Video-on-Demand, Web-TV
  - IP-Telefonie
  - Internet-Radio
  - Telefonkonferenzen, Videokonferenzen
  - Interaktive Internet-Spiele

→ **Realzeitanforderungen!!!**

Typisches Multimedia-Netzwerk im Intranet

- Beispielszenario im Home-Office



Spezielle Anforderungen

- Hohe Übertragungsraten
- Hohe Verzögerungssensibilität
- Echtzeitanforderungen

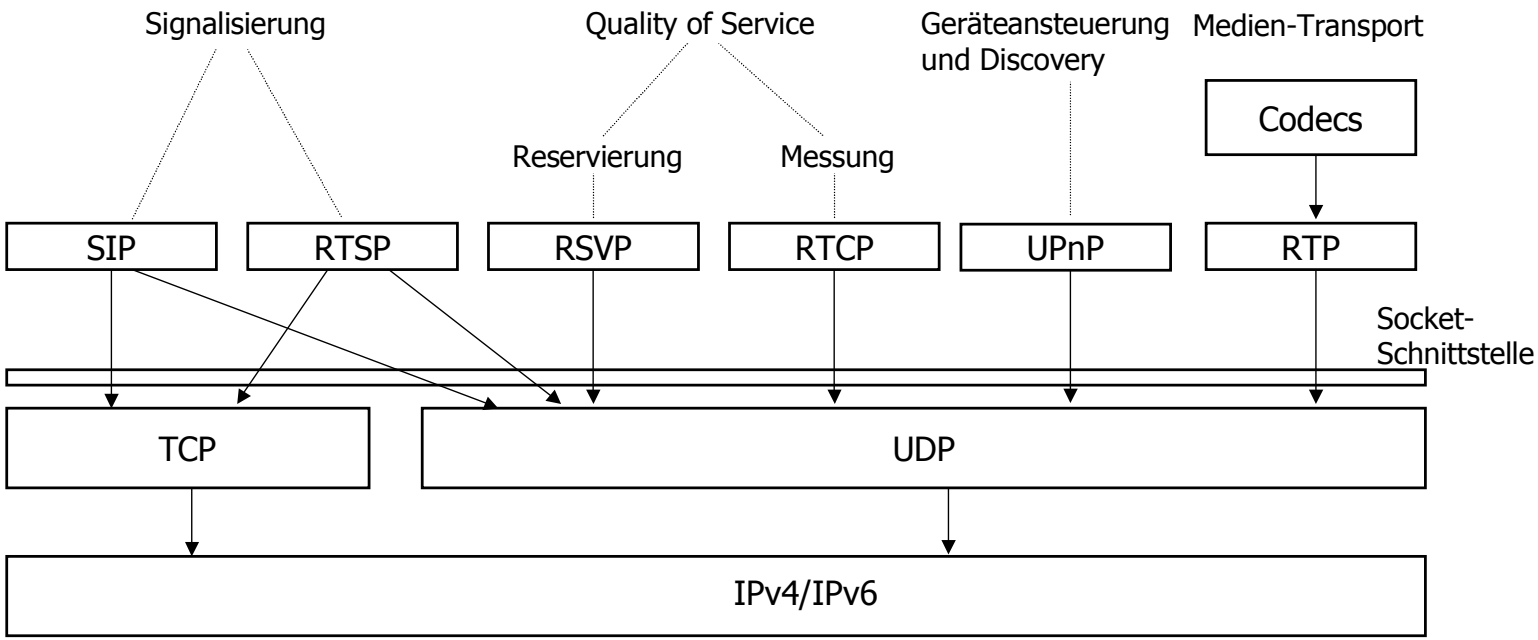
→ Komprimierungstechniken erforderlich:
  - JPEG
  - MP3
  - MPEG,…

→ Hohe Anforderungen an Quality of Service

→ Das Internetprotokoll IPv4 kann hier mit seinem Best-Effort-Ansatz nicht so viel bieten

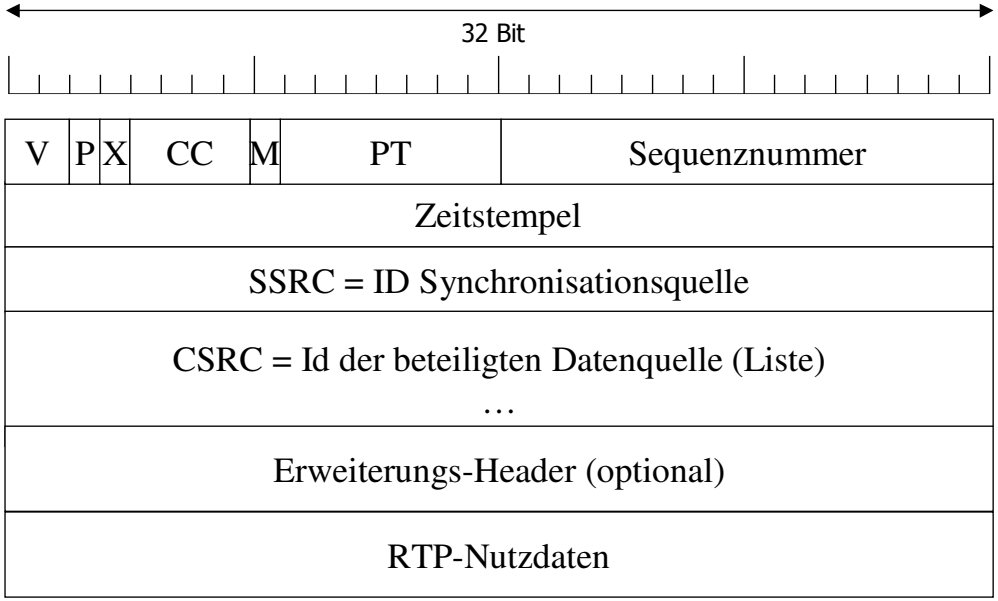
Realtime-Multimedia-Protokolle

- Überblick



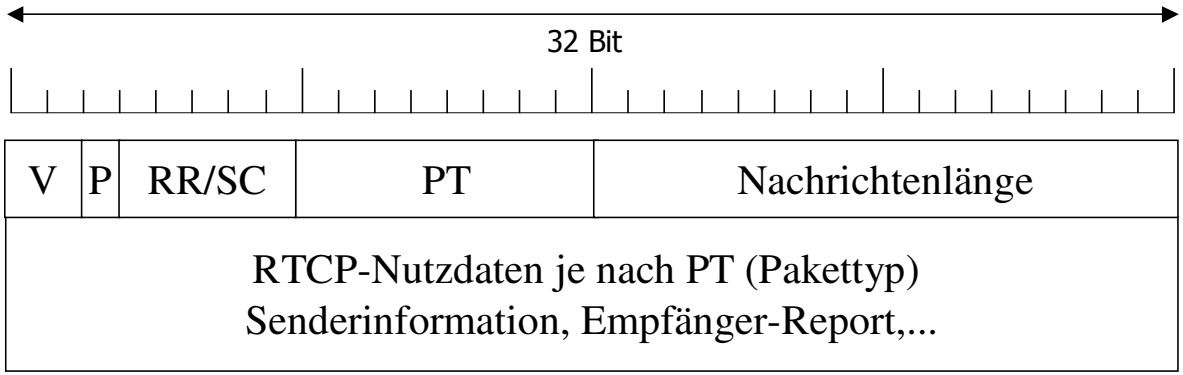
Realtime-Multimedia-Protokolle
Beispielt RTP

- RTP = Realtime Transport Protocol
- Aufgabe: „Transportprotokoll“ zur Beförderung von Medienströmen
- RFC 3550



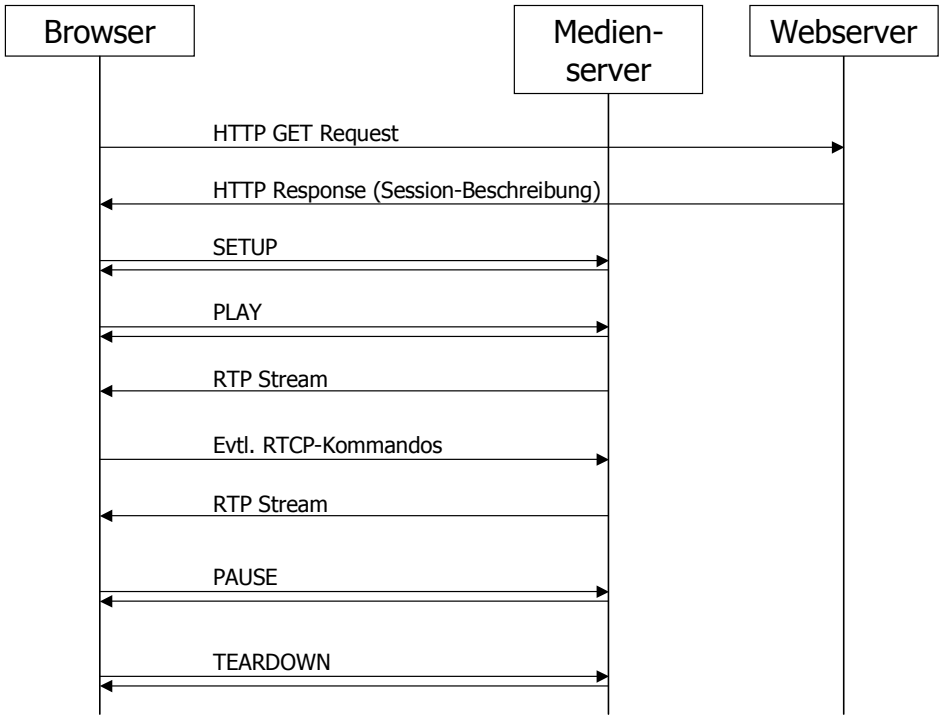
Realtime-Multimedia-Protokolle
Beispiel RTCP

- RTCP = Real Time Transport Control Protocol
- Aufgabe: Kontrollprotokoll für Medienströme
- Auch im RFC 3550 genormt



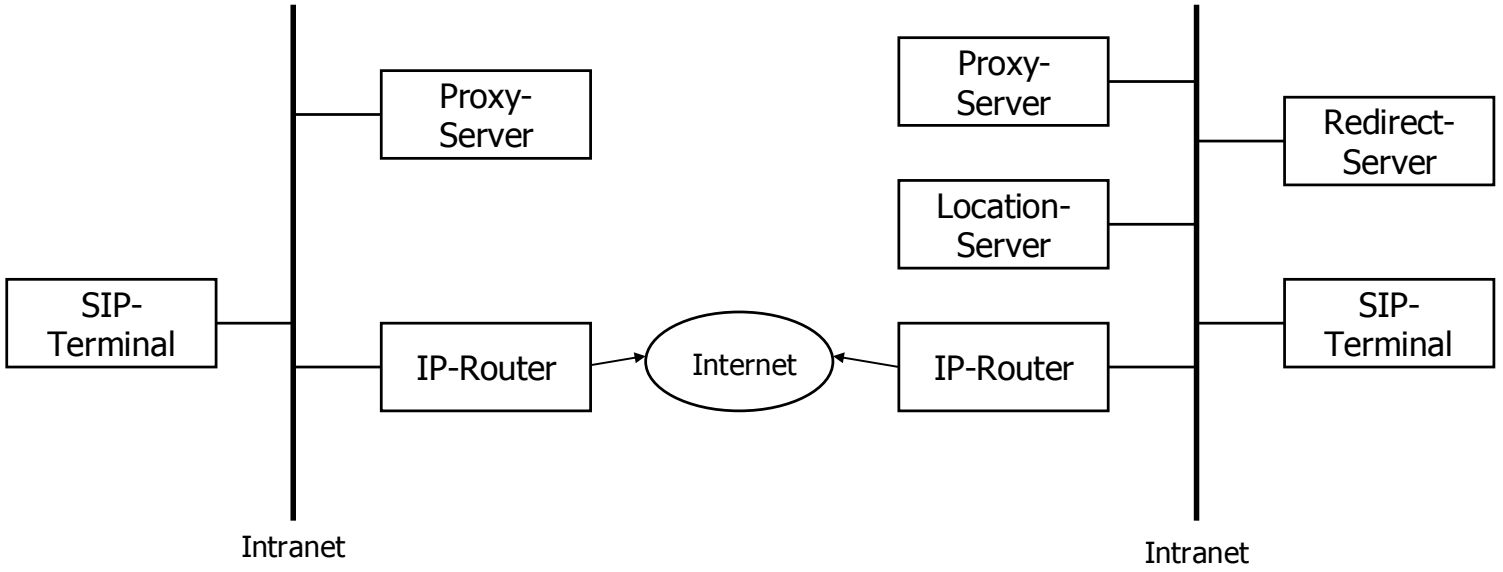
Realtime-Multimedia-Protokolle
Beispiel RTSP

- RTSP = Real-Time Streaming Protocol
- Aufgabe: Steuerung von Datenströmen (z.B. Medienströmen)
- RFC 2326



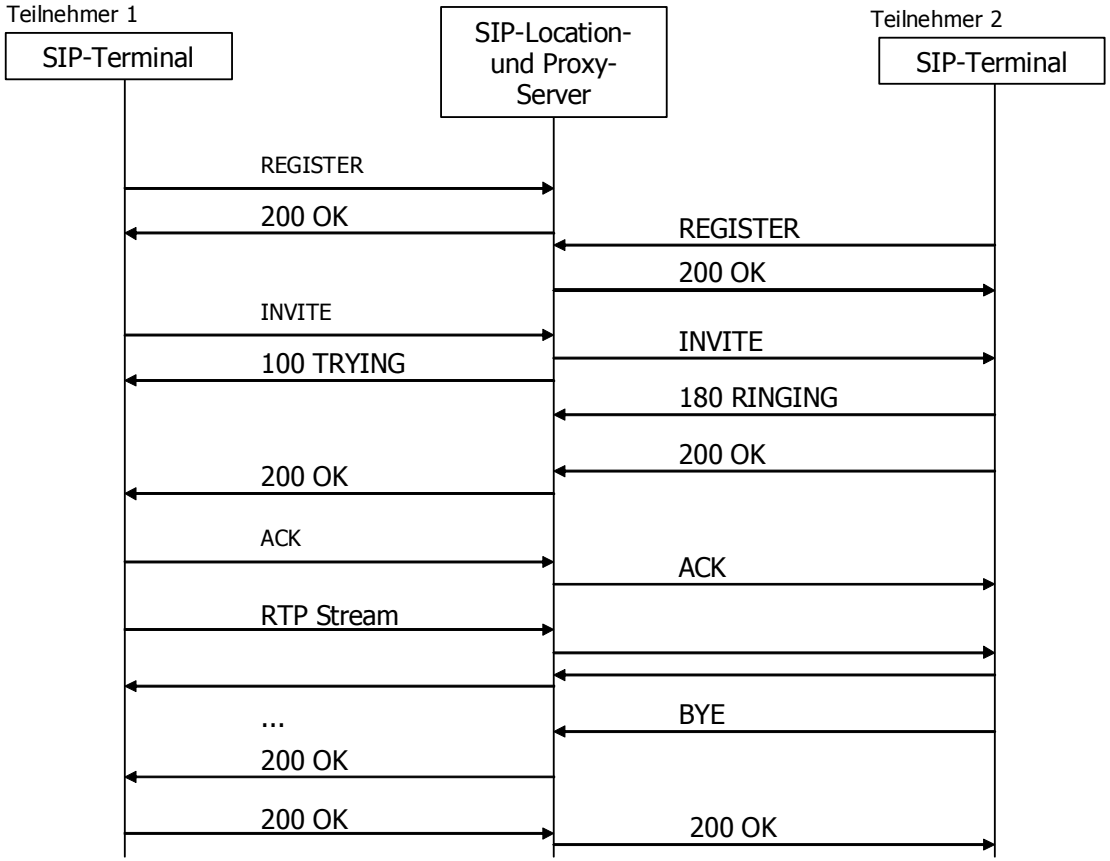
Realtime-Multimedia-Protokolle
Beispiel SIP

- Session Initiation Protocol
- Aufgabe: Aufbau und Unterhaltung von Sitzungen
- Typische Konfiguration



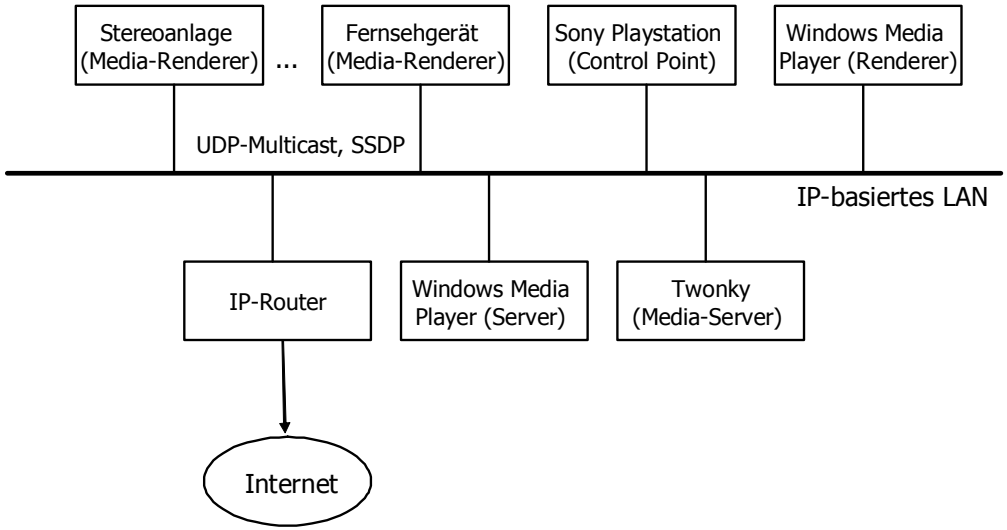
Realtime-Multimedia-Protokolle
SIP

- Typischer Ablauf einer SIP-Kommunikation



Realtime-Multimedia-Protokolle
Beispiel UPnP AV

- UPnP AV = Universal Plug & Play Audio Video
  - Von *Digital Living Network Alliance (DLNA*) unterstützt
- Beispiel einer Home-Umgebung:
  - Media-Server, Media-Renderer, Control-Point



Rückblick

1. Client-/Server versus Peer-to-Peer (P2P)
2. Domain Name System
3. HTTP
4. E-Mail
5. Multimedia-Protokolle