Vorlesung Einführung in Rechnernetze
2. Anwendungen

Prof. Dr. Martina Zitterbart

Sebastian Friebe (M.Sc.), Markus Jung (M.Sc.), Matthias Flittner (M.Sc.) , Tim Gerhard (M.Sc.)
[zitterbart | friebe | m.jung | flittner | t.gerhard]@kit.edu

Institut für Telematik, Prof. Zitterbart

Kapitel der Vorlesung

1	• Einführung
2	• Anwendungsschicht
3	• Transportschicht
4	• Vermittlungsschicht
5	• Sicherungsschicht
6	• Netzarchitektur
7	• Netzsicherheit

Überblick über Kapitel 2

2. Anwen-dungs-schicht

- 2.1 Einführung
- 2.2 Grundlagen
- 2.3 Web und HTTP
- 2.4 Electronic Mail (EMail)
- 2.5 WhatsApp
- 2.6 Domain Name System (DNS)
- 2.7 Videostreaming und CDN

Kapitel 2.1

EINFÜHRUNG

Anwendungen

- In den 70er und 80er Jahren
  - Textbasierte Anwendungen
  - EMail, Dateitransfer, Remote Access zu Computern
- 90er Jahre
  - Die Killer-Anwendung im Internet: das World Wide Web
  - Web-Surfing, Suche im Netz, E-Commerce
  - Instant-Messaging, P2P-Filesharing
- Seit 2000: steigende Vielfalt und Allgegenwärtigkeit
  - Voice over IP: Skype, Google Hangouts …
  - Nutzergenerierte Videos: You Tube, Netflix …
  - Online-Spiele: Second Life, World of Warcraft …
  - Neue soziale Netzwerke: Facebook, Instagram, Twitter …
  - Mobile Anwendungen auf Smartphone: What´s App, Pokemon Go …

Das Internet ist heute allgegenwärtig

- Nette Beispiele aus dem Alltag
  - Skype-Telefonat mit der Freundin in den USA
  - HD-Videostreaming auf dem Smartphone
  - Sprachgesteuert ein Puppenhaus bei Amazon bestellen
  - Im Flugzeug oder in der Bahn mit Bekannten chatten

- Das Internet ist zunehmend eine kritische Infrastruktur, vergleichbar mit Wasser- und Stromversorgung bzw. mit Verkehrsinfrastruktur
  - Kein Internet, kein Telefon
    - Telefonnetz ist zunehmend vom Internet abhängig
  - Kein Internet, kein Handel
    - Banken und Börse sind vom Internet abhängig
  - Kein Internet, kein Bargeld
    - Geldautomaten kommunizieren über das Internet
  - Kein Internet, leerer Supermarkt
    - Just-in-Time Delivery koordiniert über das Internet
  - …

Kapitel 2.2

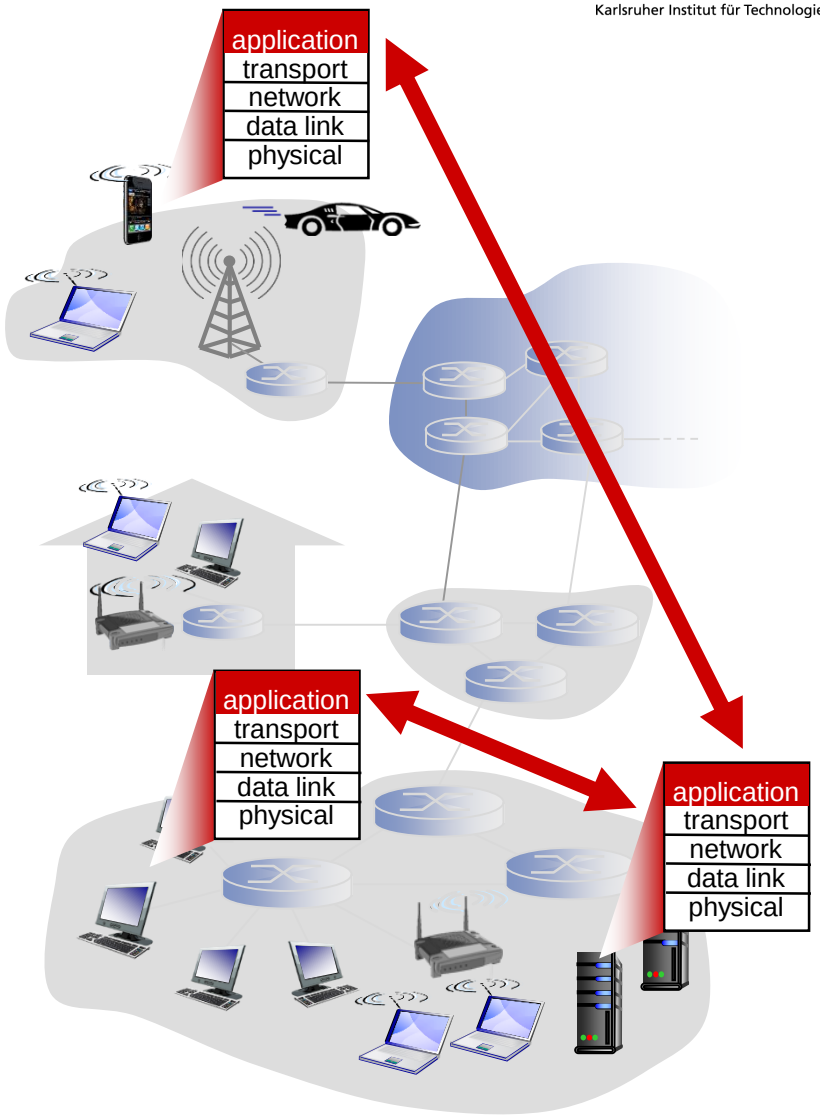
GRUNDLAGEN

Verteilte Anwendungen

- Programme
  - Laufen auf (verschiedenen) Endsystemen
  - Kommunizieren über Netze
  - Beispiel: Web-Server-Software die mit Web-Browser-Software kommuniziert

- Keine Programme für Geräte im Netz erforderlich
  - Keine Anwendungen auf Geräten im Netz

Schnelle Entwicklung und Ausbringung neuer Anwendungen möglich.



Schichtenmodell zur Strukturierung

- Kommunikation ist in Schichten organisiert
  - Anwendungsschicht oberste Schicht
    - Enthält Anwendungsprotokolle
    - Anwendung muss sich nicht um Transport der Daten kümmern
  - Transport ist Aufgabe der darunter liegenden Schichten
    - Interna sind transparent für die Anwendung
    - *Nicht* transparent für Anwendungen sind allerdings Verzögerungen (Latenzen, Delay)

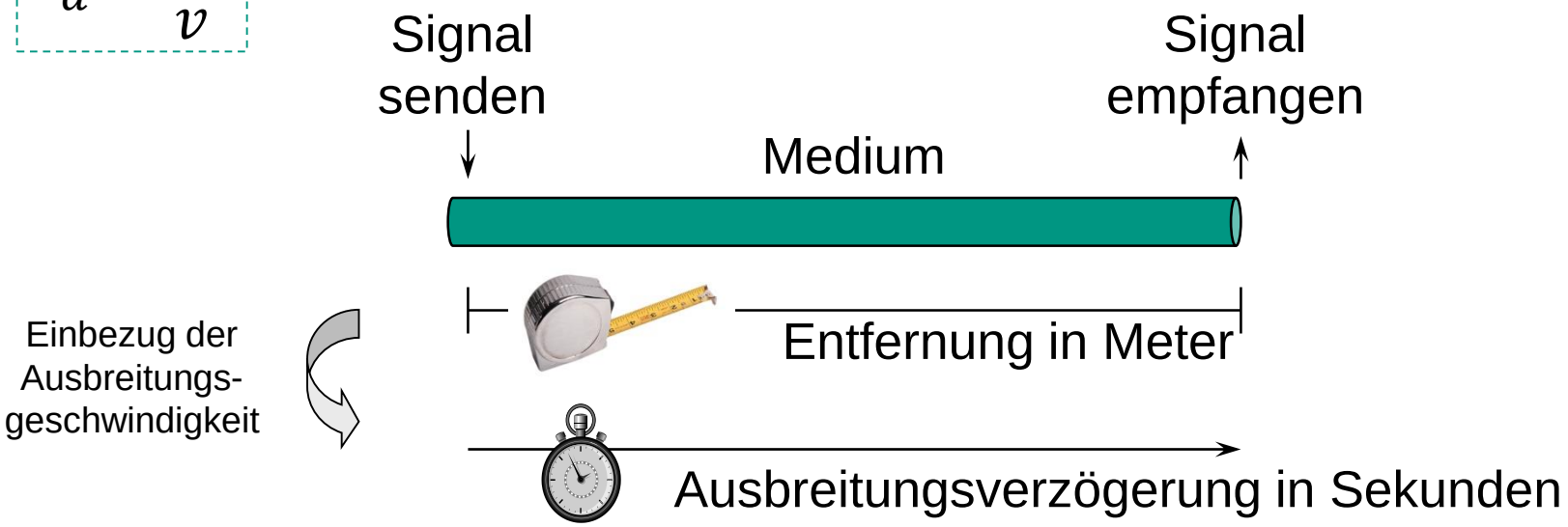
Verzögerung

- Nicht vermeidbar
  - Kann sehr unterschiedlich ausfallen
- Abhängig von
  - Ausbreitungsverzögerung t_a
  - Sendezeit t_s
  - Füllständen von Puffern

Ausbreitungsverzögerung t_a

- Zeitspanne zwischen Absenden eines Signals und dessen Eintreffen am anderen Ende des Mediums
  - Abhängig von
    - Ausbreitungsgeschwindigkeit v
      - in üblichen Medien etwa 2/3 Lichtgeschwindigkeit
    - Länge des Mediums d

$$t_a = \frac{d}{v}$$

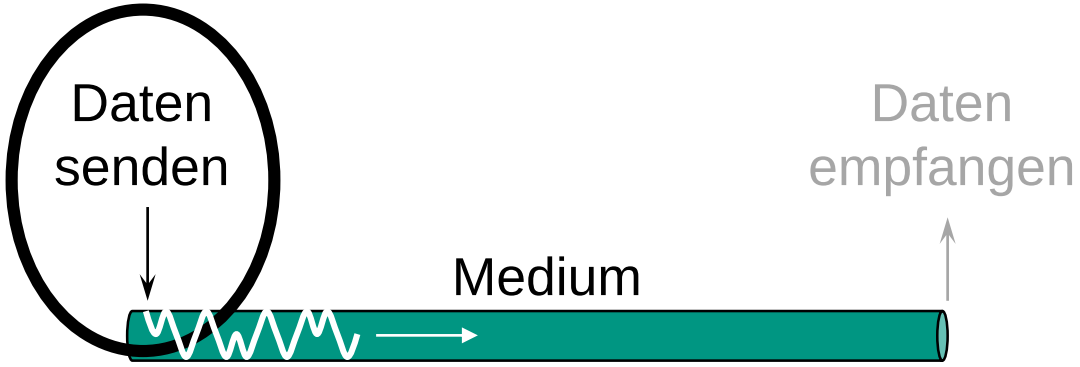


Sendezeit t_s

- Zeit zwischen Beginn und Abschluss der Sendung
  - Abhängig von
    - Datenmenge X
    - Datenrate r des Mediums (Leitung/Verbindung)
      - Anzahl der pro Zeiteinheit übertragenen Daten (i.d.R. bit/s)

$$t_s = \frac{X}{r}$$

- Achtung: Nach Abschluss der Sendung sind die Daten noch nicht beim Empfänger!
  ⇨ Ausbreitungsverzögerung



Datenraten

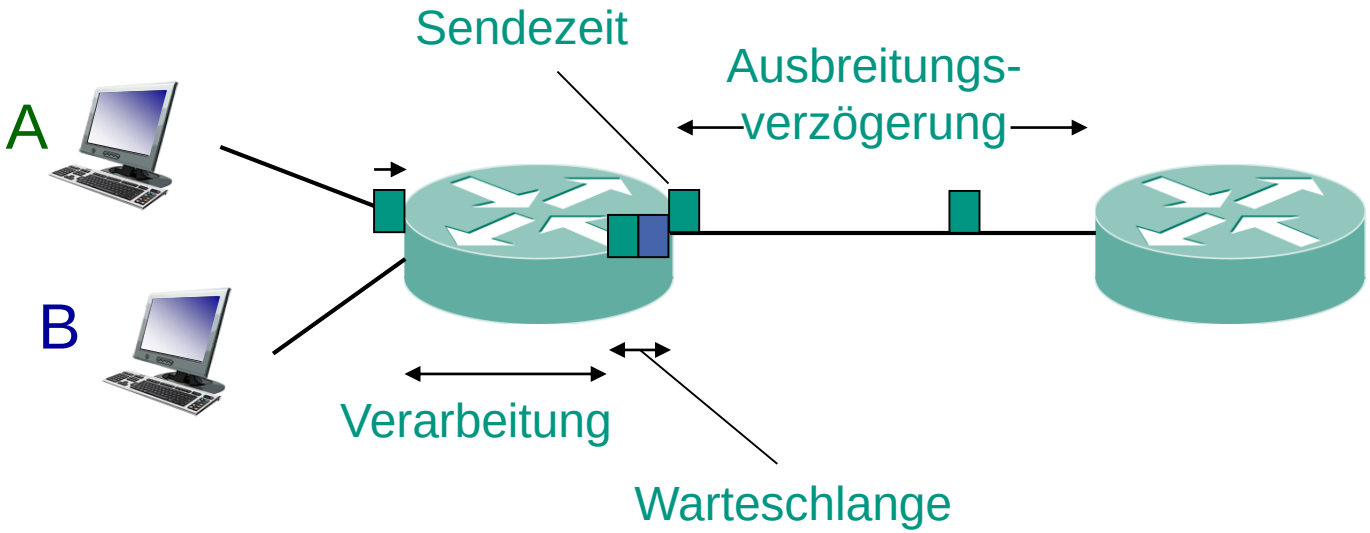
- Geschwindigkeit, mit der Daten im Netz übertragen werden
  - Typischerweise gemessen in

Einheit		Bezeichnung
bit/s		„bit pro Sekunde“ / „bit per second“
kbit/s	$1000\ \mathrm{bit/s}$	„Kilo bit pro Sekunde“
Mbit/s	$10^6\ \mathrm{bit/s}$	„Mega bit pro Sekunde“
Gbit/s	$10^9\ \mathrm{bit/s}$	„Giga bit pro Sekunde“
Tbit/s	$10^{12}\ \mathrm{bit/s}$	„Tera bit pro Sekunde“
Pbit/s	$10^{15}\ \mathrm{bit/s}$	„Peta bit pro Sekunde“

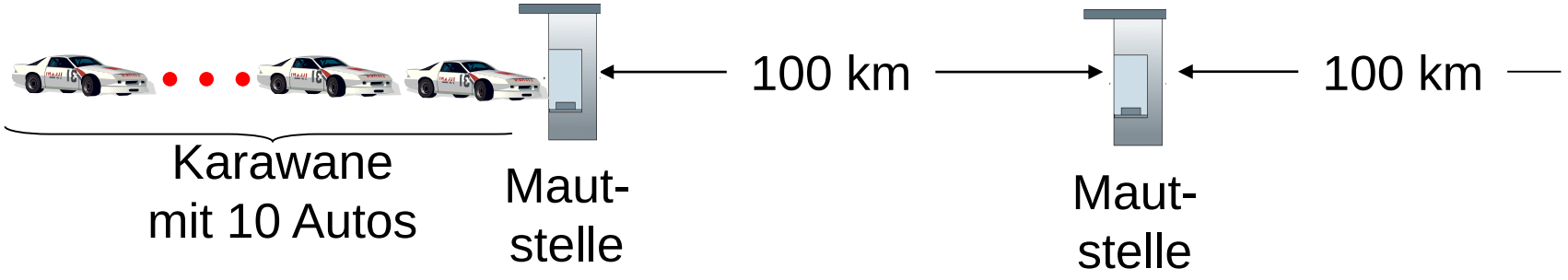
- Achtung: Stets Multiplikator (1000 vs. 1024) beachten
  - 1 kbit/s = 1000 bit/s

Verzögerungen in einem Router im Überblick

- Übertragungsdauer
  - Sendezeit + Ausbreitungsverzögerung
- Weitere Verzögerungen in einem Router
  - Pufferung der Daten in Warteschlange (Queue)
    - Warten bis Ausgangslink für Übertragung verfügbar
    - Abhängig von Last für diesen Link
  - Verarbeitung
    - Überprüfung hinsichtlich Bitfehler, Bestimmung des Ausgangslinks …
    - i.d.R. < Millisekunden



Analogon: Karawane von Autos



- „Ausbreitungsgeschwindigkeit“ der Autos: 100 km/h
- Mautstelle benötigt 12 Sekunden für Abfertigung („Sendezeit“)
- Auto ~ Bit, Karawane ~ Paket
- Frage
  - Wie lange dauert es, bis die Karawane vollständig bei der zweiten Mautstelle angekommen ist?
- Abfertigung der kompletten Karawane an der Mautstelle
  - 10*12 Sekunden
- Zeit die das letzte Auto benötigt um von Mautstelle 1 nach Mautstelle 2 zu gelangen
  - 100 km / (100 km/h) = 1 Stunde
- Antwort
  - 62 Minuten

Pingo

http://pingo.upb.de/

Pingo

http://pingo.upb.de/

Reale Verzögerungen im Internet?

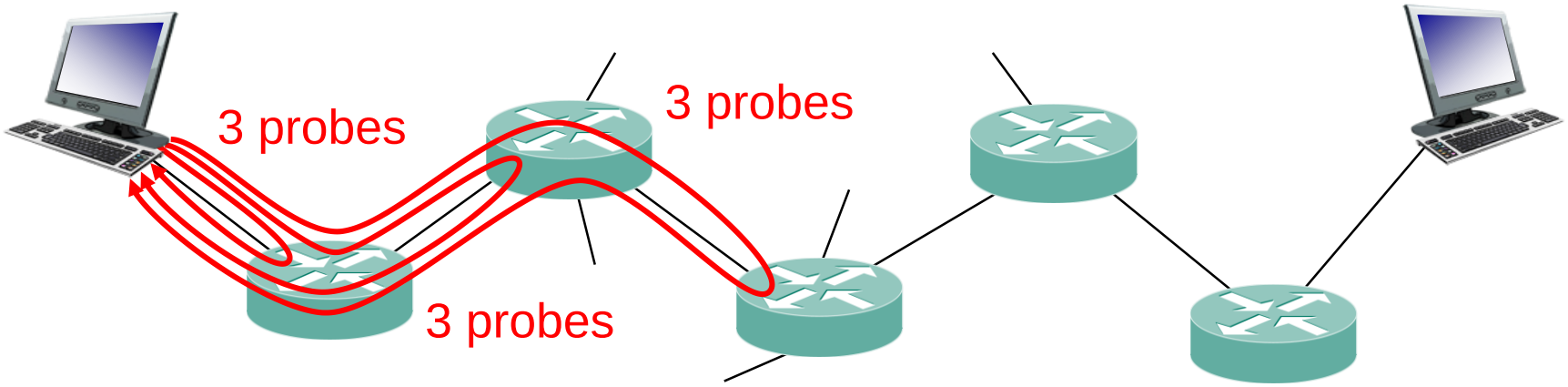
- Messen mit dem Programm traceroute
  - Misst die Verzögerung eines Pakets von der Quelle zu den Routern auf dem Weg zum Ziel
  - Für alle i (i = 1, 2, 3 …)
    - Sende drei Pakete die Router i auf dem Weg zum Ziel erreichen
    - Router i sendet Pakete zurück zur Quelle
    - Quelle misst Zeit zwischen dem Senden und dem Empfangen der Pakete

Windows:

```
>tracert kit.edu
```

Linux:

```
$ traceroute kit.edu
```



Reale Verzögerungen im Internet – Ein Beispiel

traceroute: gaia.cs.umass.edu → www.eurecom.fr

		3 Messungen		
1	cs-gw (128.119.240.254)	1 ms	1 ms	2 ms
2	border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)	1 ms	1 ms	2 ms
3	cht-vbns.gw.umass.edu (128.119.3.130)	6 ms	5 ms	5 ms
4	jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)	16 ms	11 ms	13 ms
5	jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)	21 ms	18 ms	18 ms
6	abilene-vbns.abilene.ucaid.edu (198.32.11.9)	22 ms	18 ms	22 ms
7	nycm-wash.abilene.ucaid.edu (198.32.8.46)	22 ms	22 ms	22 ms
8	62.40.103.253 (62.40.103.253)	104 ms	109 ms	106 ms
9	de2-1.de1.de.geant.net (62.40.96.129)	109 ms	102 ms	104 ms
10	de.fr1.fr.geant.net (62.40.96.50)	113 ms	121 ms	114 ms
11	renater-gw.fr1.fr.geant.net (62.40.103.54)	112 ms	114 ms	112 ms
12	nio-n2.cssi.renater.fr (193.51.206.13)	111 ms	114 ms	116 ms
13	nice.cssi.renater.fr (195.220.98.102)	123 ms	125 ms	124 ms
14	r3t2-nice.cssi.renater.fr (195.220.98.110)	126 ms	126 ms	124 ms
15	eurecom-valbonne.r3t2.ft.net (193.48.50.54)	135 ms	128 ms	133 ms
16	194.214.211.25 (194.214.211.25)	126 ms	128 ms	126 ms
17		*	*	*
18		*	*	*
19	fantasia.eurecom.fr (193.55.113.142)	132 ms	128 ms	136 ms

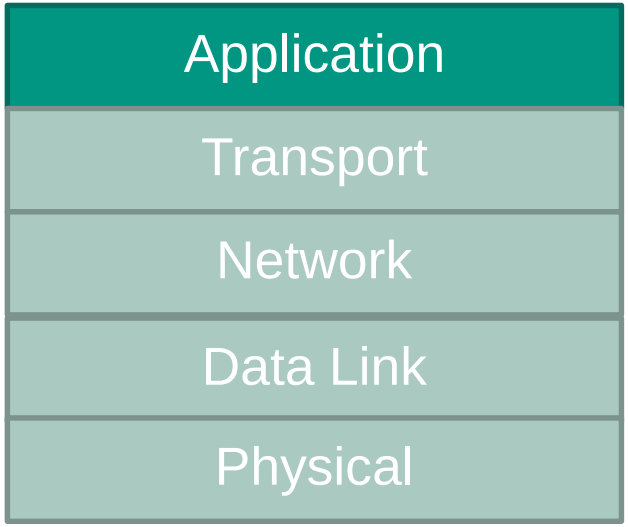
Seekabel

* Keine Antwort (Paketverlust oder Router antwortet nicht)

traceroute-Messungen aus verschiedenen Ländern und Netzen sind durchführbar mit www.traceroute.org

Internet-Protokollstack

- Application
  - SMTP, HTTP, XMPP …
- Transport
  - Datentransfer zwischen Anwendungsprozessen
  - TCP, UDP
- Network
  - Weiterleitung von Datagrammen von der Quelle zum Ziel
  - IP
- Data Link
  - Datentransport zwischen benachbarten Geräten
  - Ethernet, 802.11 (WiFi)
- Physical
  - Bits auf dem Medium



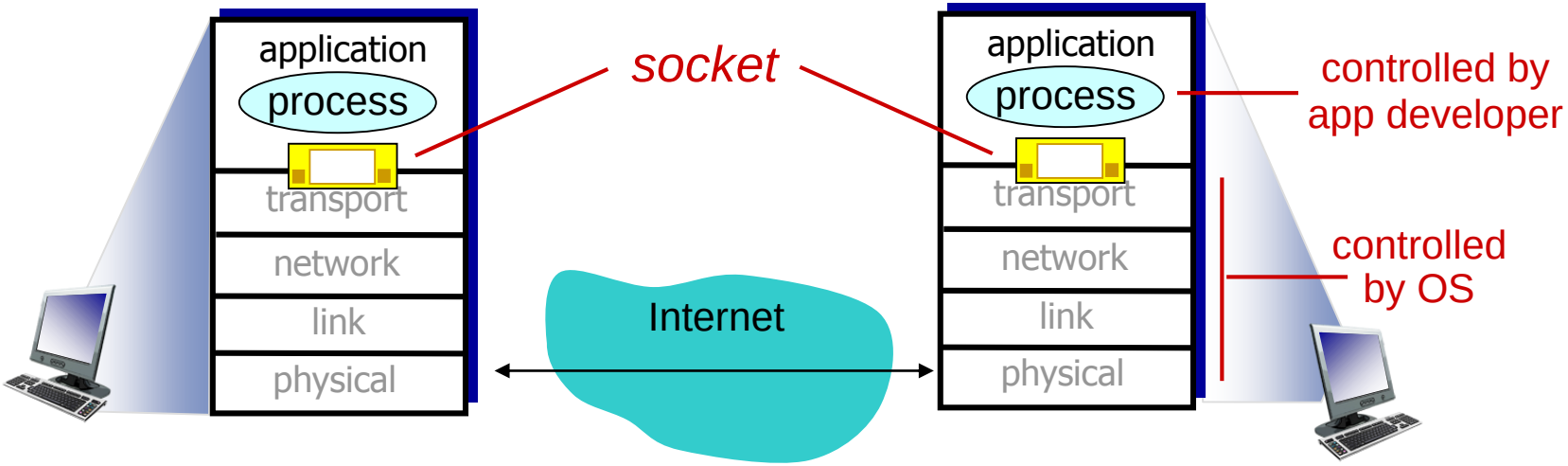
Video: IP Explained
[https://www.youtube.com/watch?v=zyL1Fud1Z1c]

Prozess, Nachricht

- Prozess bzw. Anwendungsprozess
  - Programm, das im Endsystem abläuft
  - Befindet sich in der Anwendungsschicht
  - Inter Process Communication (IPC)
    - Prozesse auf gleichem Endsystem kommunizieren, geregelt durch Betriebssystem
  - Gegenstand dieser Vorlesung
    - Prozesse auf verschiedenen Endsystemen (mit potentiell unterschiedlichen Betriebssystemen)
- Nachricht
  - Zwischen Prozessen auf unterschiedlichen Endsystemen werden Nachrichten ausgetauscht.

Socket-Interface

- Programmierschnittstelle für verteilte Anwendungen
  - Application Programming Interface (API)
  - Bereitgestellt vom Betriebssystem
  - Anwendungsprozess sendet / empfängt Nachrichten zum / vom Socket



Socket-Interface: Portnummern

- Portnummern: Multiplexing/Demultiplexing auf Endsystemen
  - Viele Prozesse auf dem Endsystem kommunizieren gleichzeitig über das Netz mit Prozessen auf entfernten Systemen
  - Ziel: Nutzer-zu-Nutzer-Kommunikation

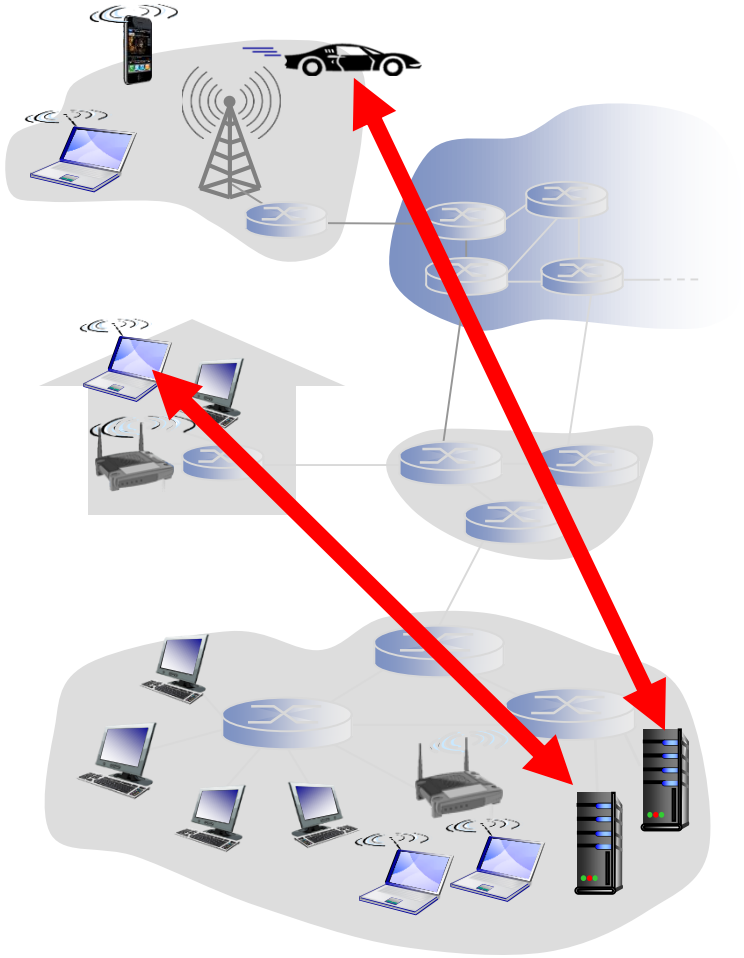
- Lösung: Sockets eindeutig identifiziert
  - Endsystem (IP-Adresse, Vermittlungsschicht)
  - Prozess auf diesem Endsystem (Portnummer, Transportschicht)

- Portnummern typischerweise standardisiert für Anwendungen
  - Port 80: Webserver
  - Port 25: Mailserver
  - Aber: Ausnahmen möglich (z.B. Webserver auf Port 4000)

Client-Server-Anwendungen

- Server
  - Ständig in Betrieb
  - Permanente IP-Adresse
  - Server häufig in Datenzentren

Data Center: „Virtueller Server“ mit vielen tausend physikalischen Servern.

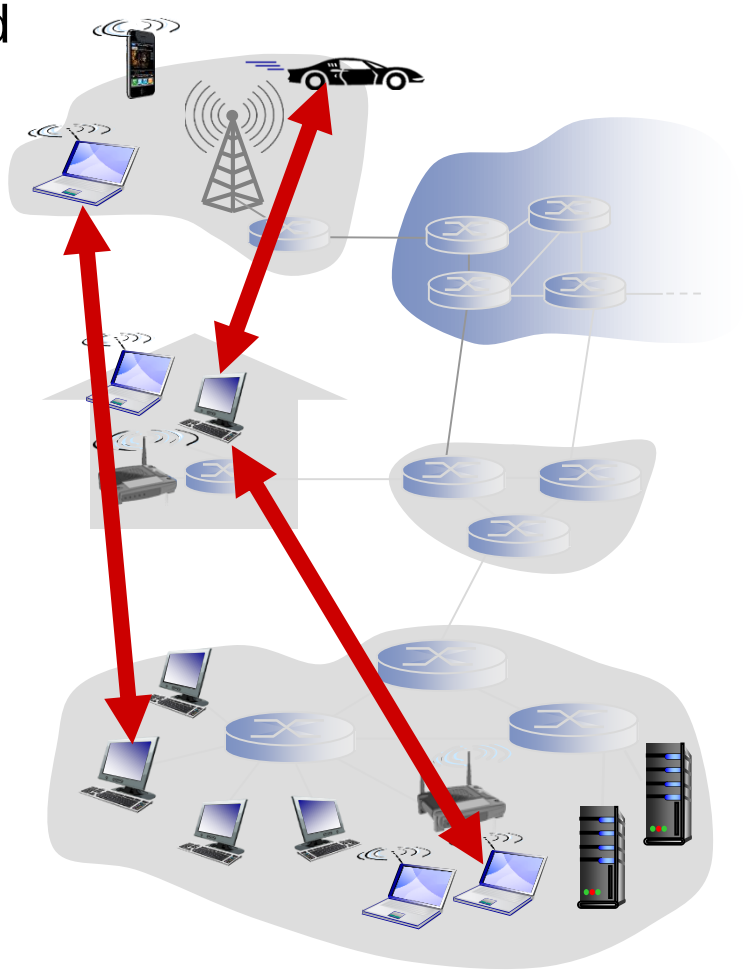


- Clients
  - Kommunizieren mit Server
  - Kommunizieren nicht direkt miteinander
  - Evtl. nicht immer verbunden
  - Evtl. mit dynamischen IP-Adressen

- Anwendungsbeispiele
  - Web
  - EMail
  - Telnet

Peer-to-Peer-Anwendungen

- Keine Server, die ständig in Betrieb sind
- Peers (Endsysteme) kommunizieren direkt miteinander
  - Fordern Dienste von anderen Peers an
  - Stellen anderen Peers Dienste zur Verfügung
  - Sind nicht permanent verbunden und wechseln dynamisch IP-Adressen
    - → komplexes Management
- Selbst-skalierend
  - Neue Peers erhöhen die Kapazität, fordern aber selbst auch Dienste an



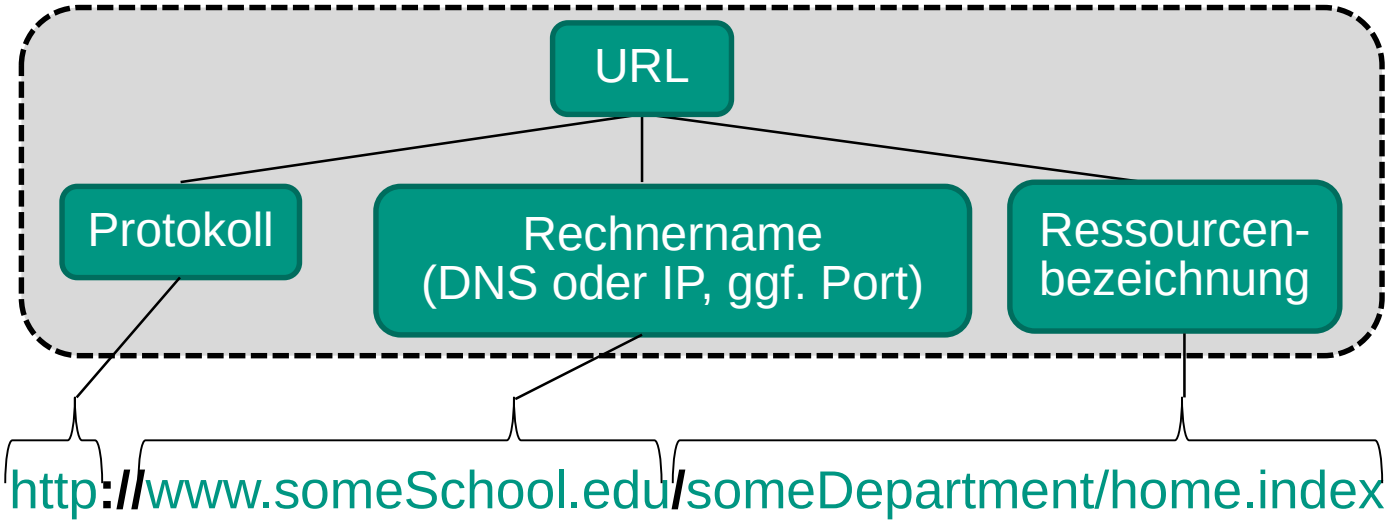
Kapitel 2.3

WEB UND HTTP

Web-Dokumente

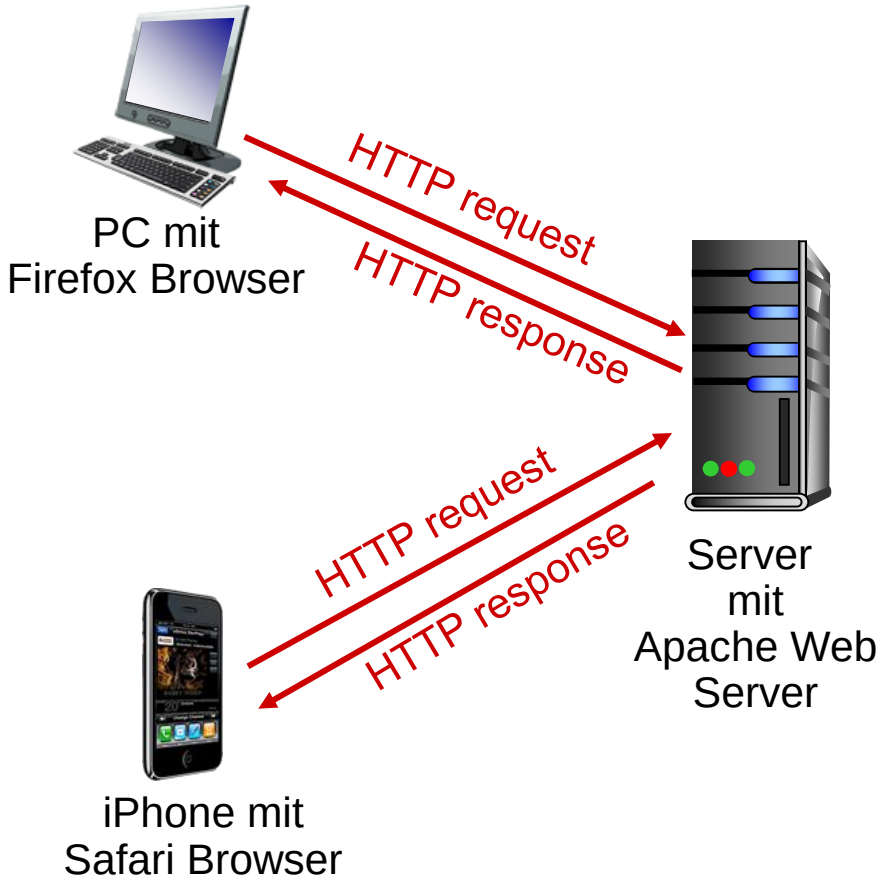
- Webseite besteht aus Basis-HTML-Datei und mehreren referenzierten Objekten
  - Z.B. PNG-Datei, JavaScript, Audiodatei, …

- Jedes Objekt ist über eine URL adressierbar
  - Uniform Resource Locator



HTTP: Überblick

- Das Protokoll der Anwendungsschicht im Web
  - HTTP: Hypertext Transfer Protocol
- Einfaches ASCII-basiertes Transferprotokoll
- Basiert auf Client/Server-Modell
  - Client
    - Browser der Web-Objekte über HTTP-Protokoll anfordert und empfängt (und darstellt)
  - Server
    - Sendet (über HTTP) angeforderte Objekte



[RFC1945]

HTTP: Überblick

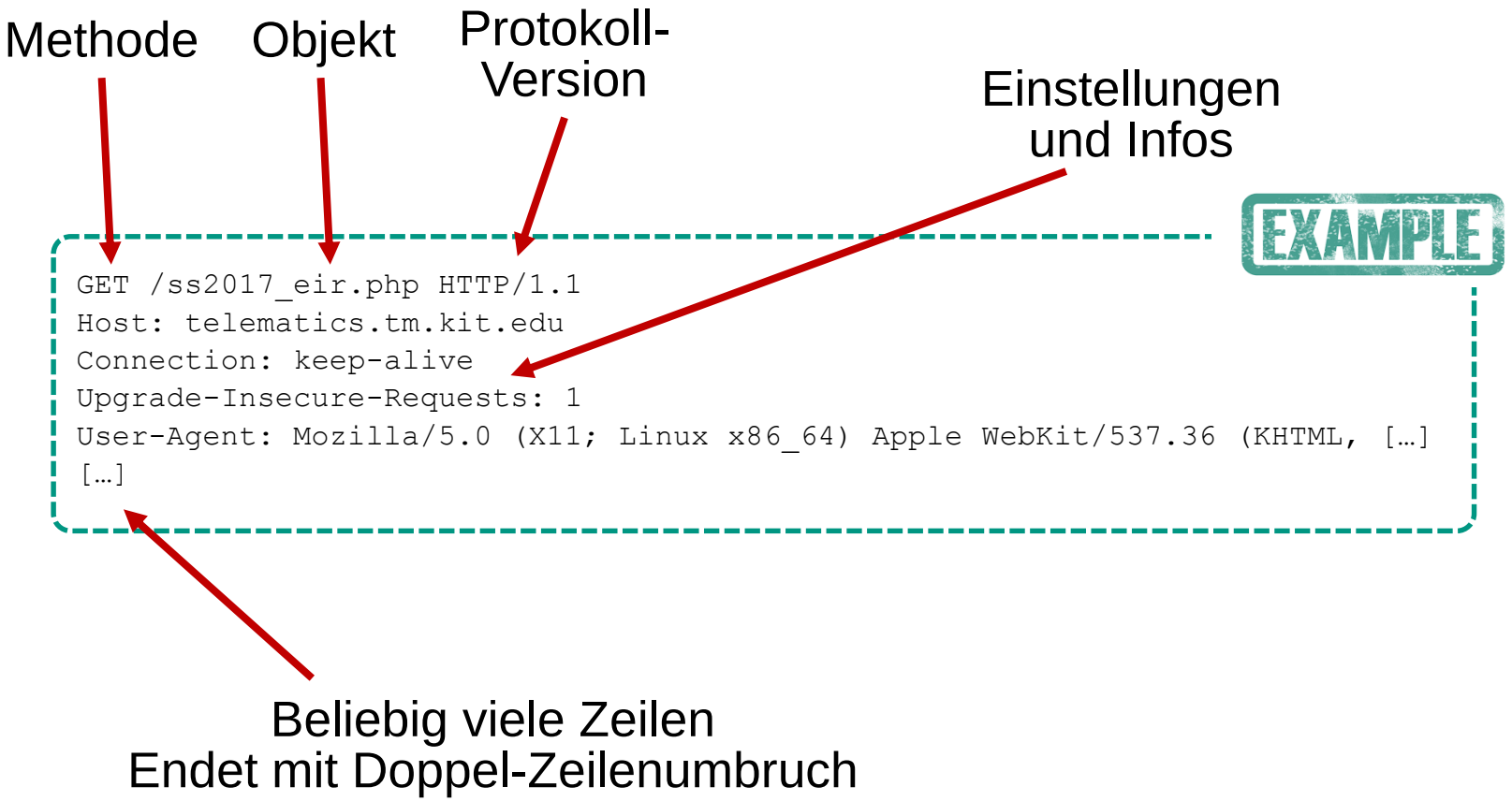
- Zwei Typen von Nachrichten
  - Request
  - Response
- Zustandslos
  - Jeder Request wird individuell bearbeitet
  - Keine Zustandsinformation auf dem Server, keine Information über frühere Requests des Clients
- Nutzt TCP zur Kommunikation
  - Client initiiert Verbindungsaufbau (Default-Port: 80)
  - Server akzeptiert Verbindung
  - Austausch von HTTP-Nachrichten
  - Abbau der TCP-Verbindung

HTTP-Methoden

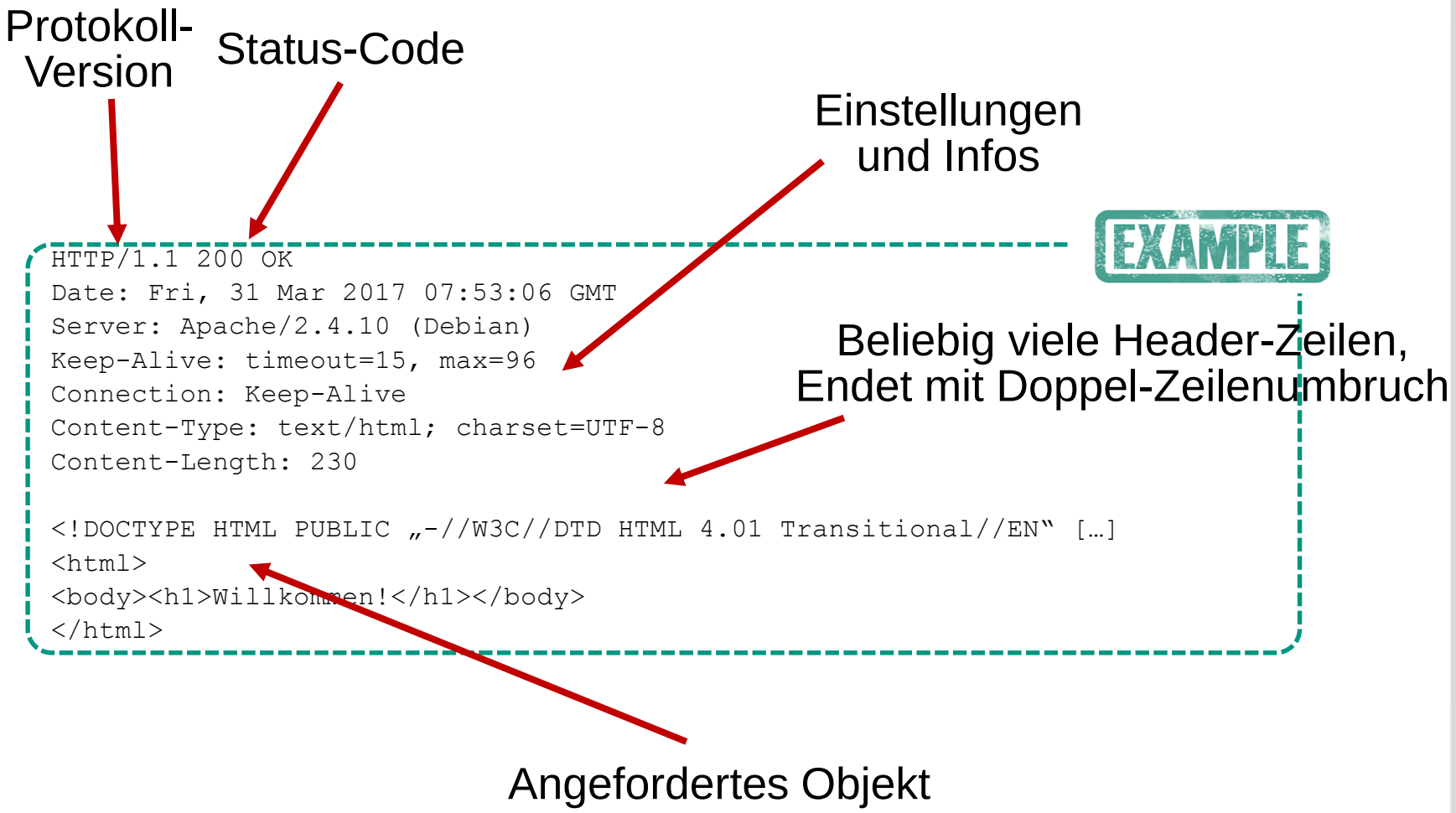
- HTTP-Anfragen können verschiedene Methoden nutzen
  - GET
    - Ressource vom Server zum Client übertragen
    - Z.B.: Normaler Webseitenaufruf
  - POST
    - Daten zu einer Ressource übertragen
    - Antwort wie GET
    - Z.B.: Formular auf Webseiten
- Weitere Methoden
  - PUT
    - Neue Ressource anlegen
  - DELETE
    - Ressource löschen
  - HEAD
    - Wie GET, aber nur HTTP-Header übertragen
  - …


```
POST /form.php HTTP/1.1
Host: aperture-science.com
Connection: keep-alive
Content-length: 20
Content-type: application/x-www-form-urlencoded

test=19&success=true
```

HTTP-Nachrichten: Request



HTTP-Nachrichten: Response



HTTP: Status Codes

- Indikator für die Verarbeitung
  - Erfolg/Fehlschlag
  - Gründe
- Beispiele
  - 200 OK
    - Erfolg; Antwort ist in dieser Nachricht
  - 301 Moved Permanently
    - Angefragtes Objekt wurde verschoben
    - Neue URL in dieser Nachricht spezifiziert
  - 400 Bad Request
    - Server hat Anfrage nicht verstanden
  - 404 Not Found
    - Angefordertes Objekt existiert nicht
  - 505 HTTP Version Not Supported

HTTP-Verbindungen

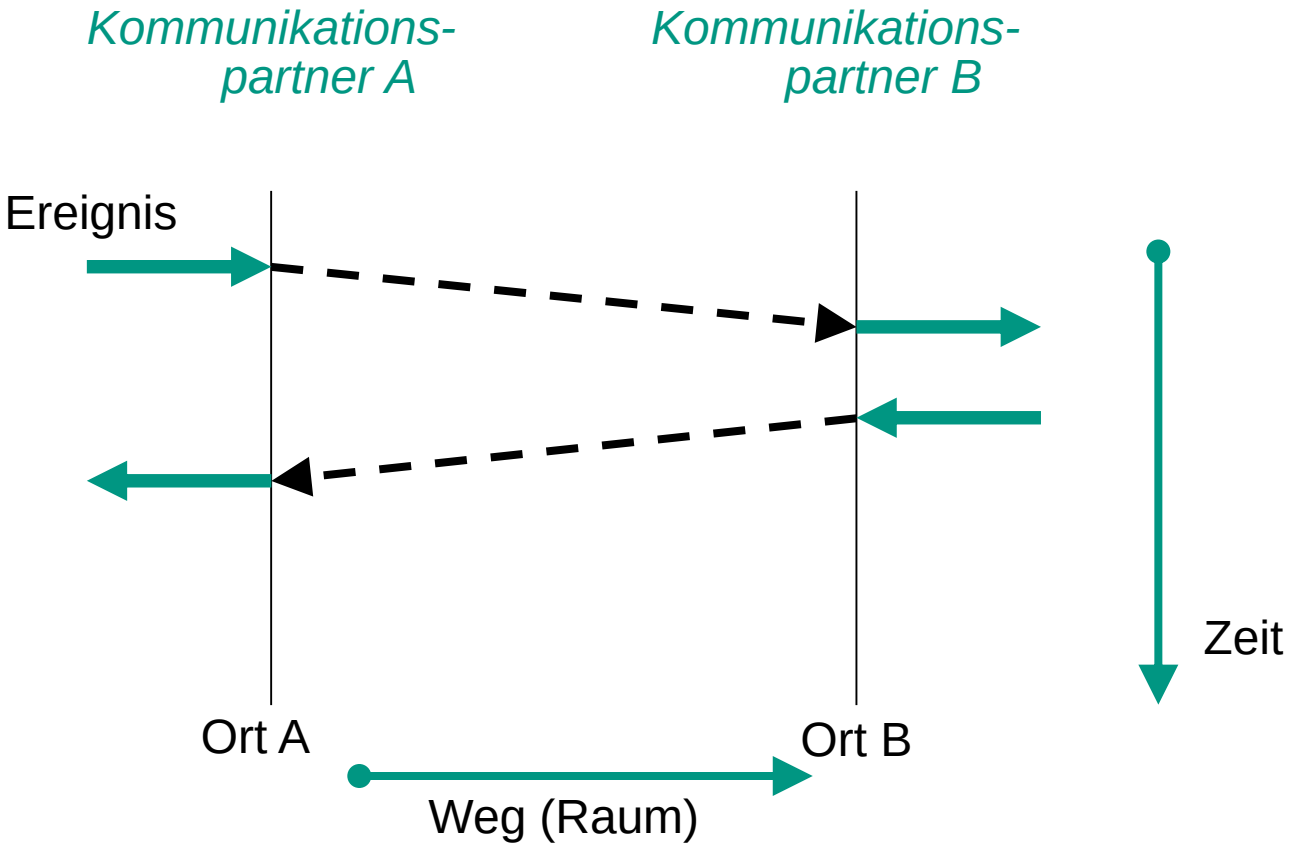
- Non-persistent HTTP
  - Über eine TCP-Verbindung wird höchstens ein Objekt gesendet
  - TCP-Verbindung wird danach geschlossen
  - Folge: Herunterladen mehrerer Objekte erfordert mehrere TCP-Verbindungen nacheinander
- Persistent HTTP
  - Über eine TCP-Verbindung können mehrere Objekte versendet werden

Einschub: Weg-Zeit-Diagramme

- Aufgabe
  - Beschreibung räumlich verteilter Abläufe
  - Grafische Darstellung innerhalb eines 2-Achsen-Koordinatensystems
    - Vertikale Achse: Zeitachse
      - Abstrahiert vom tatsächlichen Zeitpunkt
    - Horizontale Achse: Räumliche Distanz
      - Abstrahiert von der tatsächlichen räumlichen Distanz
- Dargestellter Ablauf
  - Menge von Ereignissen
    - Aufgetragen nach Ort und relativem Zeitpunkt
- Darstellung von Alternativen
  - In Weg-Zeit-Diagrammen unübersichtlich
  - Je Sachverhalt separates Diagramm

Einschub: Weg-Zeit-Diagramme

- Beschreibungsmethode zur Darstellung von Abläufen in Kommunikationssystemen
  - Vorwiegend für einfachere Sachverhalte geeignet



Non-persistent HTTP: Beispiel

Nutzer gibt folgende URL ein:

`www.someSchool.edu/someDepartment/home.index`

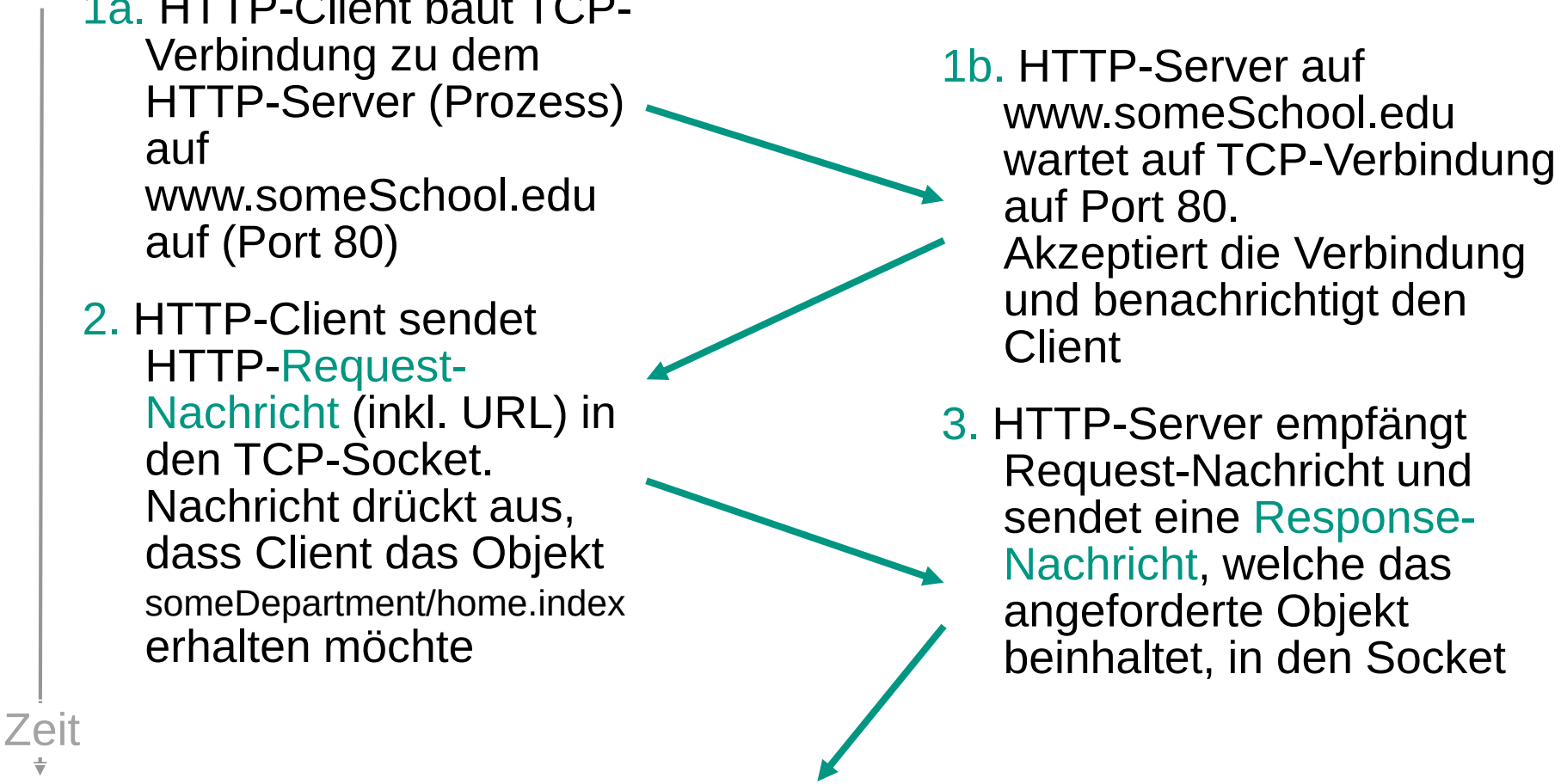
(beinhaltet Text und referenziert 10 JPEG-Bilder)

1a. HTTP-Client baut TCP-Verbindung zu dem HTTP-Server (Prozess) auf www.someSchool.edu auf (Port 80)

1b. HTTP-Server auf www.someSchool.edu wartet auf TCP-Verbindung auf Port 80. Akzeptiert die Verbindung und benachrichtigt den Client

2. HTTP-Client sendet HTTP-Request-Nachricht (inkl. URL) in den TCP-Socket. Nachricht drückt aus, dass Client das Objekt someDepartment/home.index erhalten möchte

3. HTTP-Server empfängt Request-Nachricht und sendet eine Response-Nachricht, welche das angeforderte Objekt beinhaltet, in den Socket



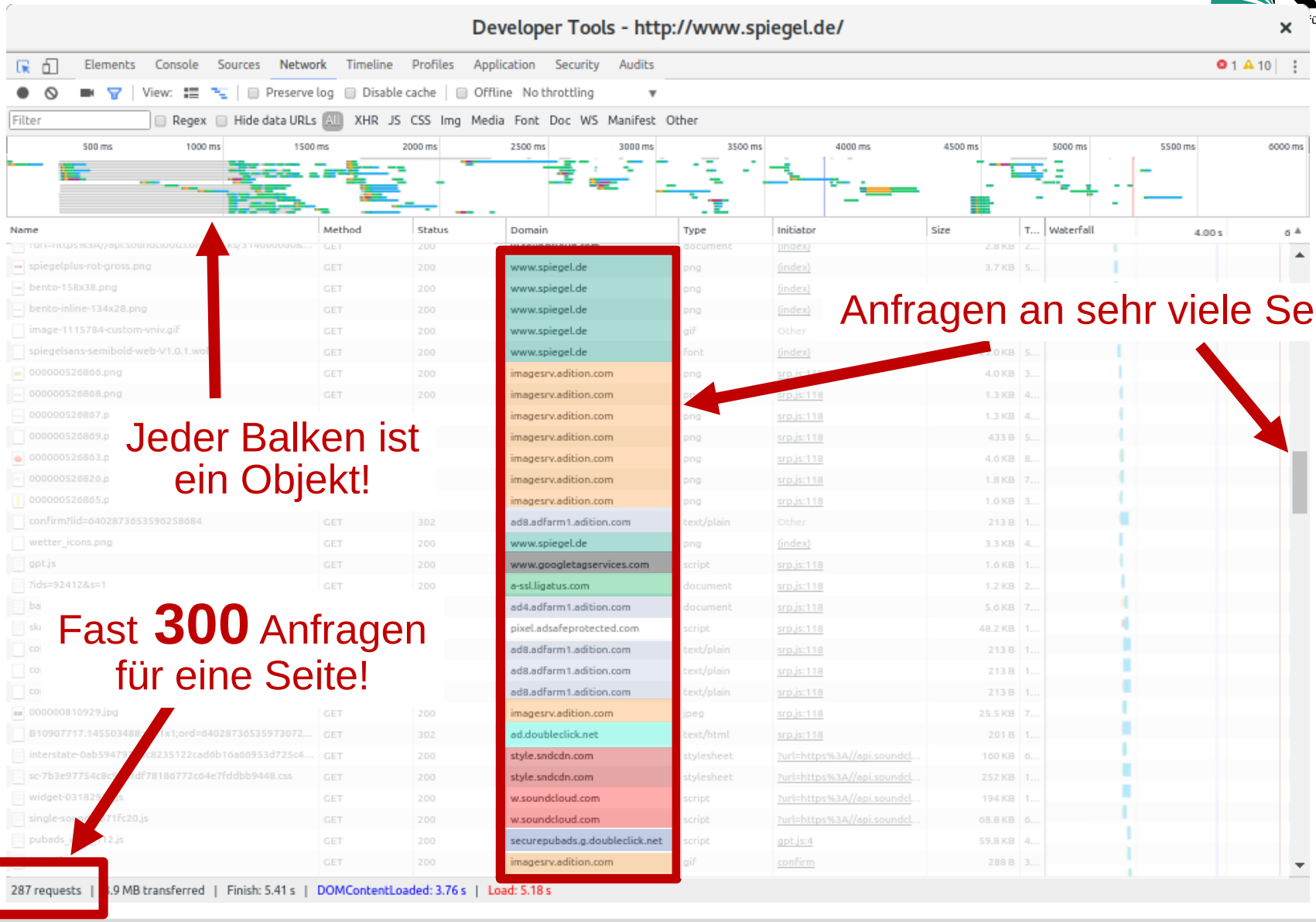
Non-persistent HTTP (cont.)

4. HTTP-Server schließt TCP-Verbindung

5. HTTP-Client empfängt Response-Nachricht, die die HTML-Datei enthält, und zeigt sie an. Dabei findet er 10 referenzierte JPEG-Objekte. HTTP-Client schließt TCP-Verbindung.
6. Schritte 1-5 werden für jedes JPEG-Objekt wiederholt.

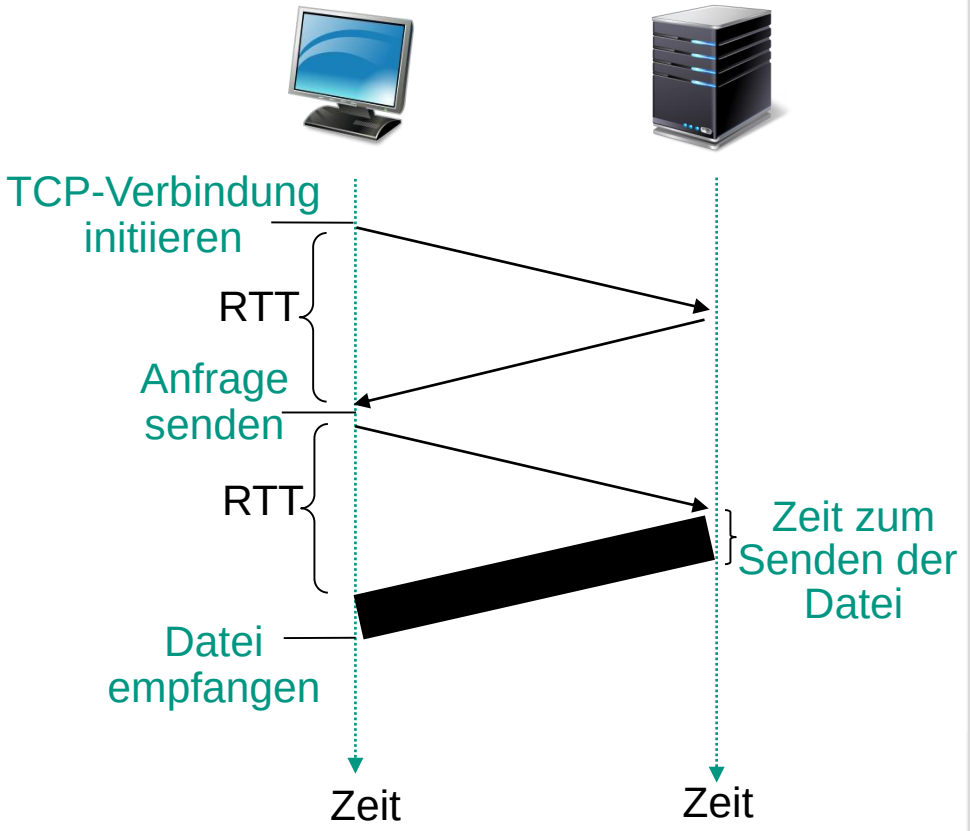
Zeit

Beispiel: Objekte auf Spiegel.de



Non-persistent HTTP: Antwortzeit

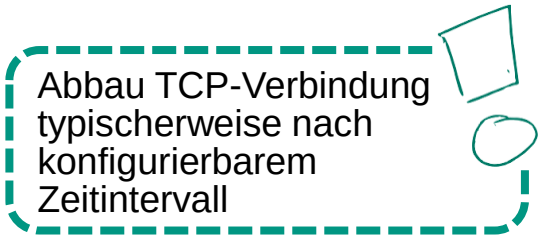
- Round-Trip-Time (RTT)
  - Zeit, die ein Paket vom Sender zum Empfänger und zurück benötigt
- HTTP-Antwortzeit
  - Eine RTT für Verbindungsaufbau
  - Eine RTT für HTTP-Anfrage und erste Bytes der Antwort
  - Zeit t_s für das Senden der Datei
  - Antwortzeit: 2•RTT + t_s



Persistent HTTP

- Non-persistent HTTP
- Benötigt >2 RTT pro Objekt
- Betriebssystem-Overhead für jede TCP-Verbindung
- Browser nutzen oft parallele TCP-Verbindungen um alle Objekte zu laden

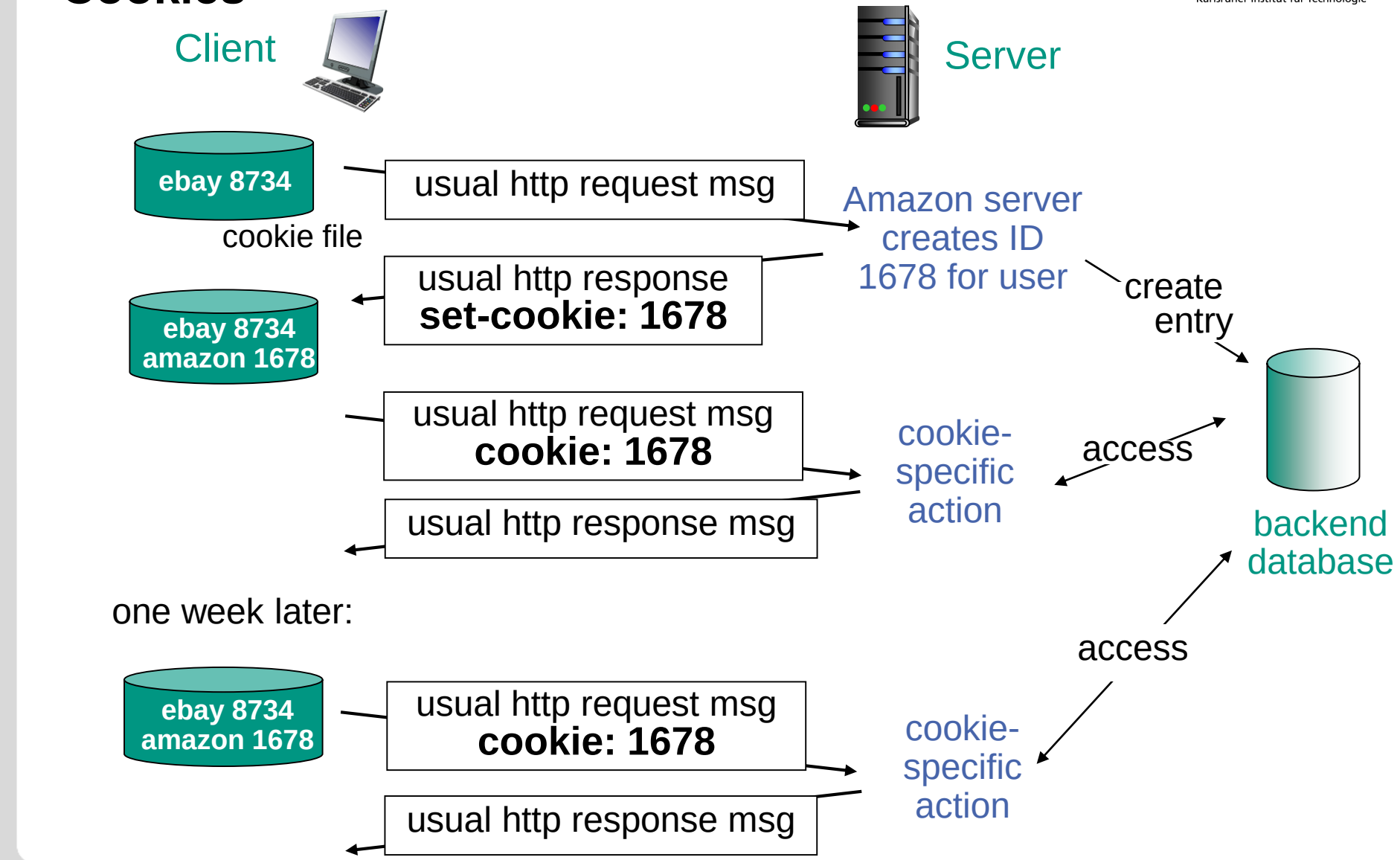
- Persistent HTTP
- Server hält TCP-Verbindung nach Senden offen
- Aufeinanderfolgene HTTP-Nachrichten zwischen selben Server und Client können über eine TCP-Verbindung gesendet werden
- Client fragt an, sobald er ein referenziertes Objekt entdeckt
- Nur eine RTT pro referenziertem Objekt



Cookies

- Nutzer-Server-Zustand
  - Server kann Inhalt abhängig von der Nutzeridentifikation bereitstellen
  - Z.B. für
    - Autorisierung
    - Einkaufswagen
    - Empfehlungen
  - Die meisten kommerziellen Webseiten nutzen heute Cookies
    - Nutzereinstellungen
    - Nutzerauthentifizierung
- Vier Komponenten
  - Cookie-Information in der HTTP-Response-Nachricht
  - Diese Cookie-Information wird in nachfolgenden HTTP-Request-Nachrichten genutzt
  - Datei mit Cookies auf dem Endsystem des Nutzers. Wird vom Browser verwaltet
  - Datenbank bei der Webseite. Server muss Cookies richtig interpretieren können

Cookies


Client
Server
ebay 8734
cookie file
usual http request msg
Amazon server creates ID 1678 for user
create entry
usual http response
set-cookie: 1678
ebay 8734
amazon 1678
usual http request msg
cookie: 1678
cookie-specific action
access
backend database
usual http response msg
one week later:
access
ebay 8734
amazon 1678
usual http request msg
cookie: 1678
cookie-specific action
usual http response msg

Cookies: Beispiel

Client 

 Server

EXAMPLE

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: userid=56378; Expires=Wed, May 10 2017 08:00:00 GMT
Set-Cookie: setting_pagecolor=red

[HTML doc]
```

EXAMPLE

```
GET /article/783748
Host: www.server.com
Cookie: userid=56378; setting_pagecolor=red
```

Cookies und Privacy

- Durch Cookies können Webseiten Nutzer unterscheiden
  - Externe Werbeanbieter können Nutzer über sehr viele Webseiten tracken
- Dadurch können Webseiten und Werbetreibende sehr viel über Ihre Nutzer lernen
- Könnte durch Nutzereingriff umgangen werden, aber
  - Webbrowser weisen ihre Nutzer nicht auf Cookies hin
  - Dadurch sind Cookies für die meisten Nutzer unsichtbar

Kapitel 2.4

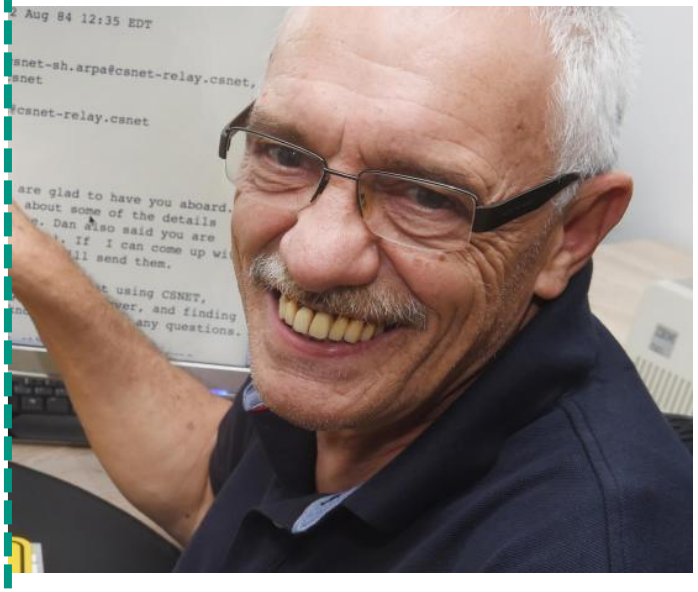
ELECTRONIC MAIL (E-MAIL)

Karlsruhe: Erste E-Mail in Deutschland

EXAMPLE

Erste E-Mail in Deutschland

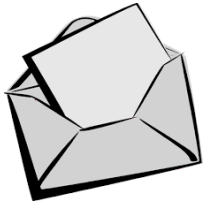
- 3. August 1984
  - Zustellung hat 24 Stunden gedauert
- Empfänger:
  Michael Rotert <rotert@germany>
  - Informatikrechnerabteilung
    Universität Karlsruhe (TH)
- Absender: Laura Breden
  - Cambridge, Massachusetts
- Inhalt: Wir sind froh, euch an Bord zu haben!



[https://de.wikipedia.org/wiki/E-Mail#Geschichte]

EMail

- Vergleichbar mit traditioneller Post

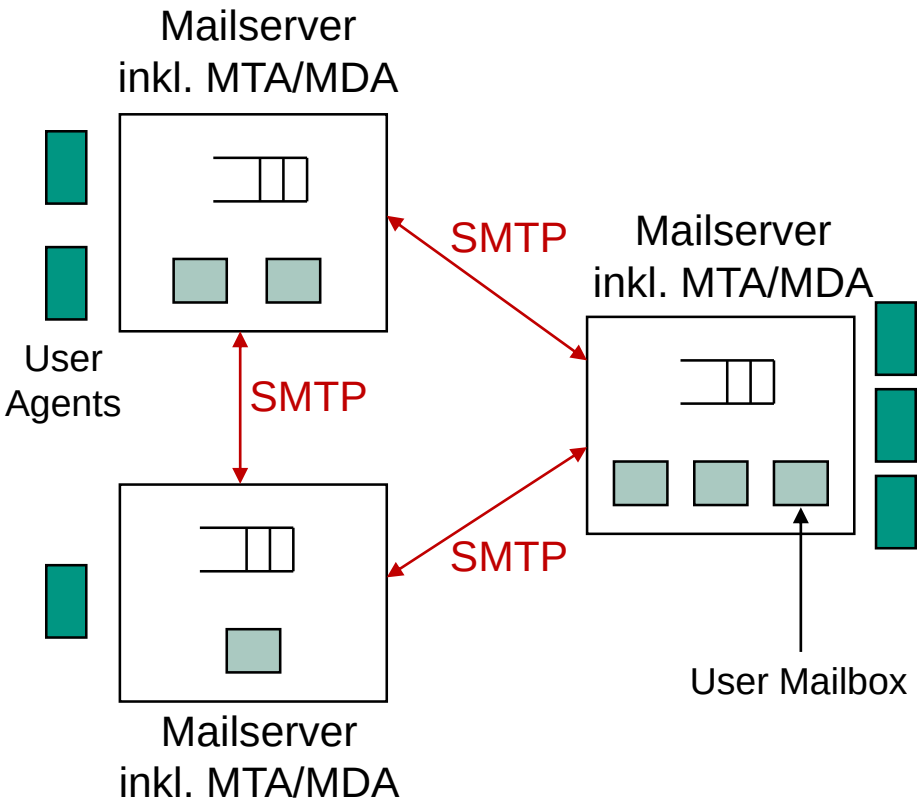


- Asynchroner Datenaustausch zwischen Benutzern

- Inhalte
  - Text, Bilder, Audio, Video und Dateien

Grundlegende Komponenten

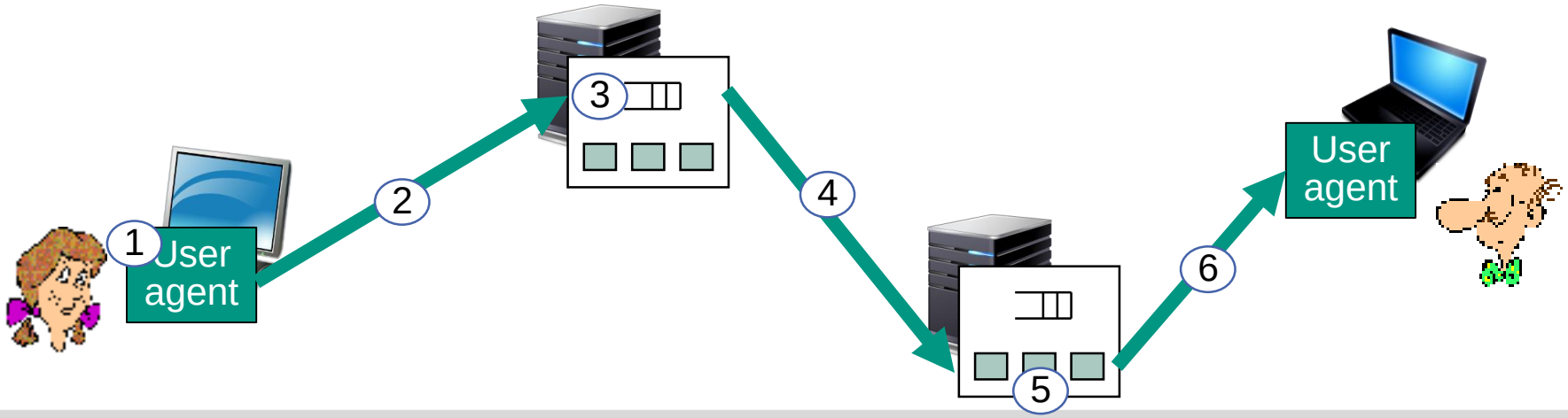
- User Agent (UA)
  - Lesen, senden, weiterleiten …
  - Beispiele: Outlook, Thunderbird
- Mail Server
  - Mail Transfer Agent (MTA)
  - Mail Delivery Agent (MDA)
  - Mailboxen der Nutzer
    - Verwaltung der E-Mails pro Nutzer
- Simple Mail Transfer Protocol (SMTP)
  - Client/Server
  - Transfer der E-Mail vom UA zum Mailserver
  - Transfer von E-Mails zwischen Mailservern



[RFC2821]

Szenario: Alice sendet Nachricht zu Bob

- Alice verfasst E-Mail an bob@someschool.edu mit UA
- Alices UA sendet die Nachricht an ihren Mail Server (SMTP)
- Alices Mailserver reiht Nachricht in Warteschlange ein und öffnet TCP-Verbindung zu Bobs Mailserver
- Alices Mailserver sendet Nachricht an Bobs Server (SMTP)
- Bobs Mailserver speichert Nachricht in Bobs Mailbox
- Bobs UA ruft Nachricht ab (POP3 oder IMAP)



Simple Mail Transfer Protocol (SMTP)

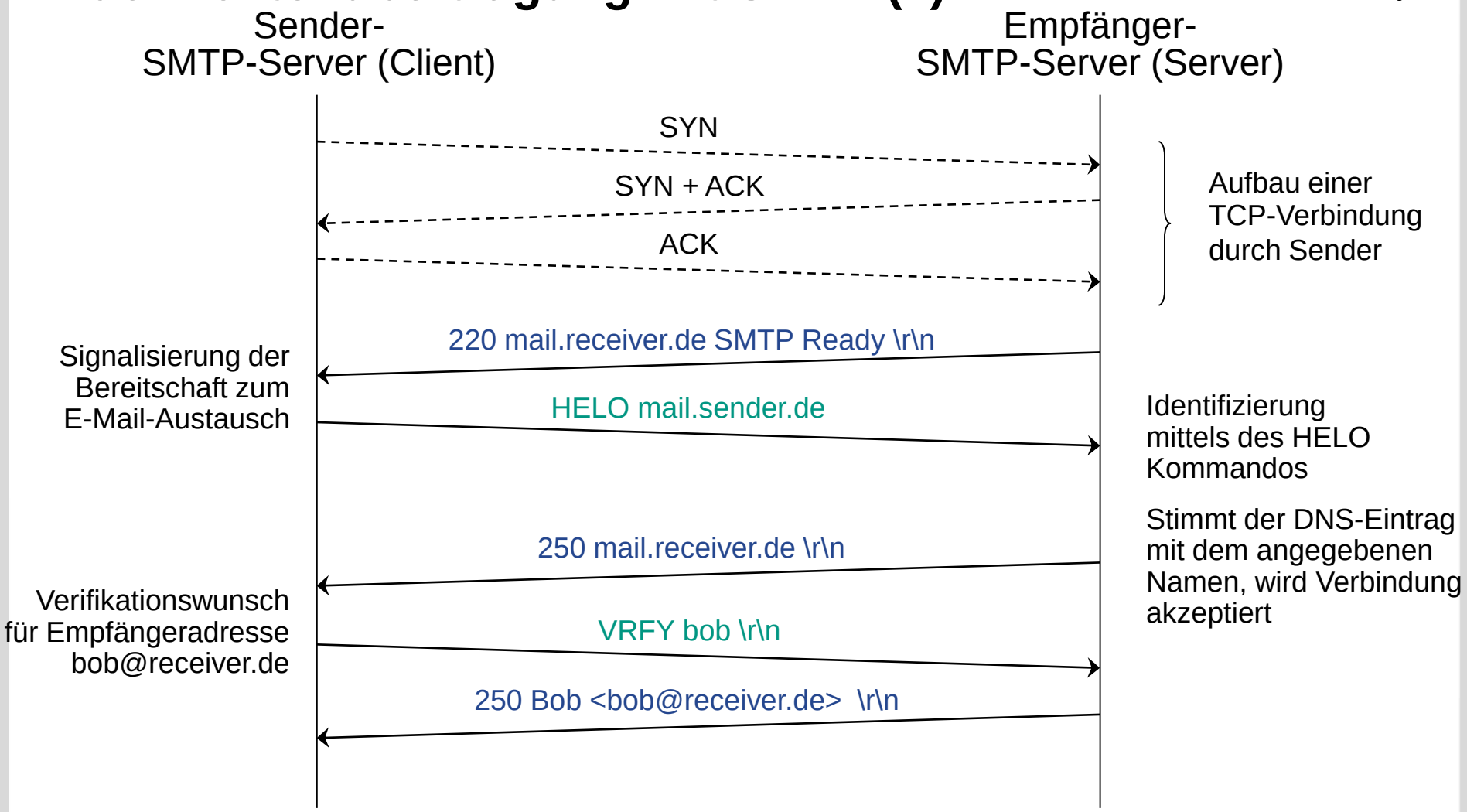
- SMTP arbeitet in drei Übertragungsphasen
  - Handshake
  - Übermittlung von Nachrichten
  - Abschluss
- Command/Response-Interaktionen
  - Ähnlich Request/Response bei HTTP
  - Kommandos: ASCII-Text
  - Antwort: Statuscode und –nachricht
- Nachrichten müssen 7-bit-ASCII sein
- Nutzt TCP
  - Zuverlässige Übertragung von E-Mails zum Server, Port 25

HTTP: Pull-Protokoll

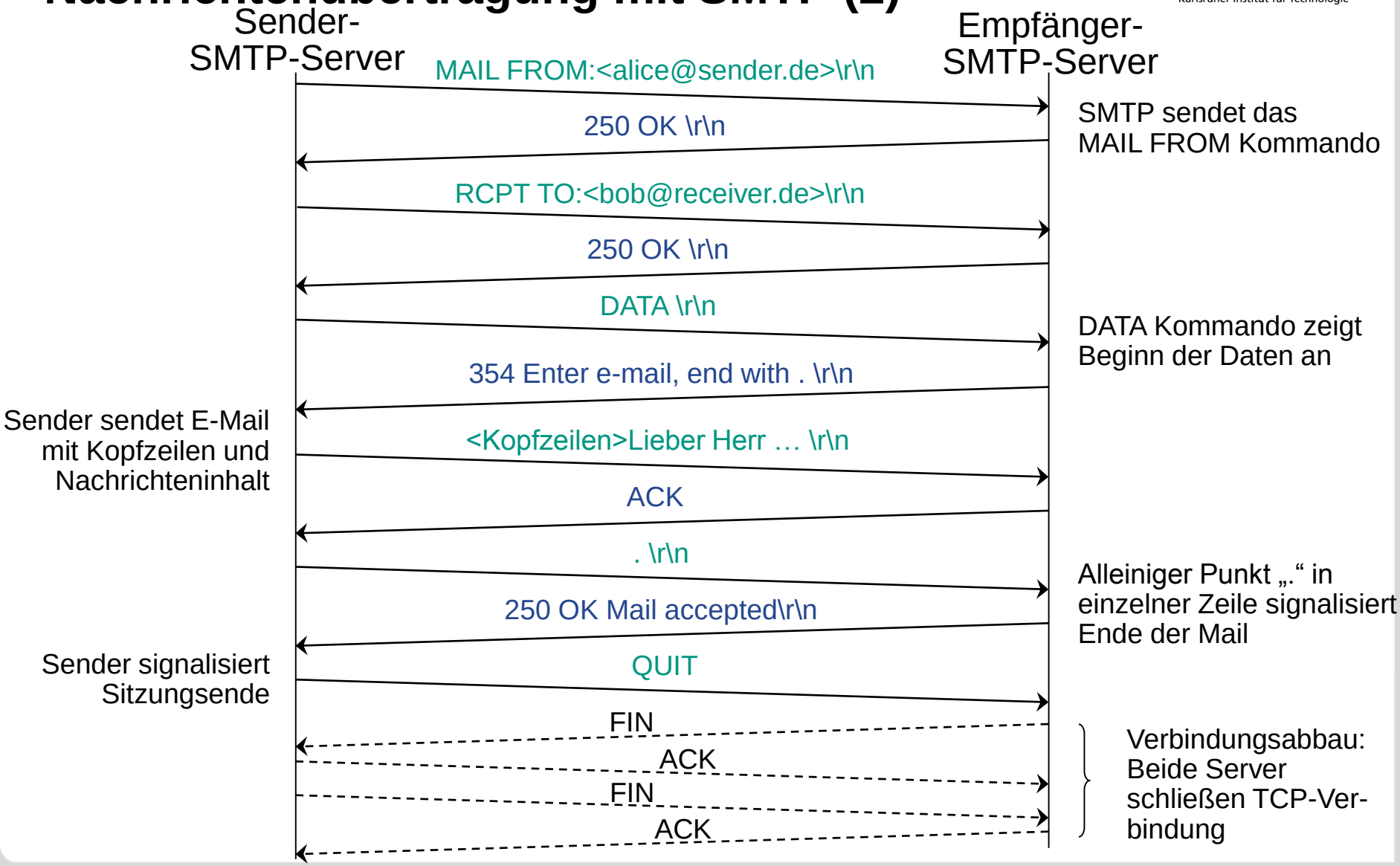
SMTP: Push-Protokoll

Zum Server, nicht zum adressierten Benutzer.

Nachrichtenübertragung mit SMTP (1)



Nachrichtenübertragung mit SMTP (2)



E-Mail-Format nach RFC 822

EXAMPLE

Header

Body

```
Envelope-to: mroehr@tm.uka.de
Received: from i72pc108.tm.uni-karlsruhe.de ([141.3.71.162] helo=tm.uka.de)
          by irams1.ira.uka.de with smtp (Exim 3.30 #7 (Debian))
          id 161oxI-0005j9-00
          for <mroehr@tm.uka.de>; Thu, 08 Nov 2001 14:11:07 +0100
Message-Id: <E161oxI-0005j9-00@irams1.ira.uka.de>
From: zit@tm.uka.de
To: mroehr@tm.uka.de
Date: Thu, 08 Nov 2001 14:11:07 +0100

Hallo!
...
```

- Grundlegender Aufbau: Header + Body
- Zusätzliche Header-Felder
  - BCC: Blind Carbon Copy
  - Reply-To: Antwort-Adresse falls nicht an Sender-Adresse zurück geschickt werden soll
  - In-Reply-To: Message-ID auf die Bezug genommen wird
- Probleme
  - Keine Übertragung von Binärdateien oder ausführbaren Programmen möglich
  - Keine länderspezifischen Zeichen (7-bit ASCII)

[RFC822]

MIME – Multipurpose Internet Mail Extensions

- Problem: SMTP sieht nur einfache ASCII-Texte (7-bit ASCII) als Nachrichten vor

[RFC2045/2046/6532]

- MIME
  - Erweitert den Kopfteil einer Nachricht um Formatinformation
  - Content-Type: Definiert den Typ des E-Mail-Inhalts
    - Bsp.: content-Type: text/plain; charset=ISO-8859-1
  - Content-Transfer-Encoding: Definiert die Transfer-Syntax, in der die Daten des Hauptteils übertragen werden
    - Bsp.: content-transfer-encoding: base64
- Transfersyntax nach RFC2045
  - 7bit
  - 8bit
  - Binary
  - Base64
  - Quoted-printable

MIME: Transfer-Syntax

- Transfersyntax Quoted-printable erlaubt nationale Sonderzeichen
  - Kodierungsvorschrift für Bytes außerhalb des 7-Bit-ASCII-Zeichensatzes
    - Ermöglicht so Abwärtskompatibilität zur 7-Bit-ASCII-Kodierung
  - Kodierung der Zeichen außerhalb der dezimalen Bereiche 33-60 und 62-126
    - Vorangestelltes „=“-Zeichen plus Hexadezimal-Code des Bytes
    - z.B. Darstellung von André (im ISO-8859-1 Zeichensatz): Andr=E9
- Transfersyntax Base64 bildet Binärdaten auf lesbare ASCII-Zeichen ab
  - Drei aufeinanderfolgende Textbytes (24 Bit) werden mit vier 6-Bit-Werten kodiert
    - 64 Zeichen (a-z, A-Z, 0-9, +, /)
  - ca. 30% Overhead durch Base64-Kodierung
- Darstellung kodierter Wörter
  - =? charset ? encoding ? encoded-text ?=
- Beispiel aus RFC 1522

```
From: =?US-ASCII?Q?Keith_Moore?= <moore@cs.utk.edu>
To: =?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld@dkuug.dk>
CC: =?ISO-8859-1?Q?Andr=E9_?= Pirard <PIRARD@vm1.ulg.ac.be>
Subject: =?ISO-8859-1?B?SWYgeW91IGNhbiByZWFkIHRoaXMgeW8=?=
=?ISO-8859-2?B?dSB1bmRlcnN0YW5kIHRoZSBleGFtcGxlLg==?=
```

```
From: Keith Moore <moore@cs.utk.edu>
To: Keld Jørn Simonsen <keld@dkuug.dk>
CC: André Pirard <PIRARD@vm1.ulg.ac.be>
Subject: If you can read this you understand the example.
```

`?Q?` steht für Quoted-printable Kodierung

`?B?` steht für Base64 Kodierung

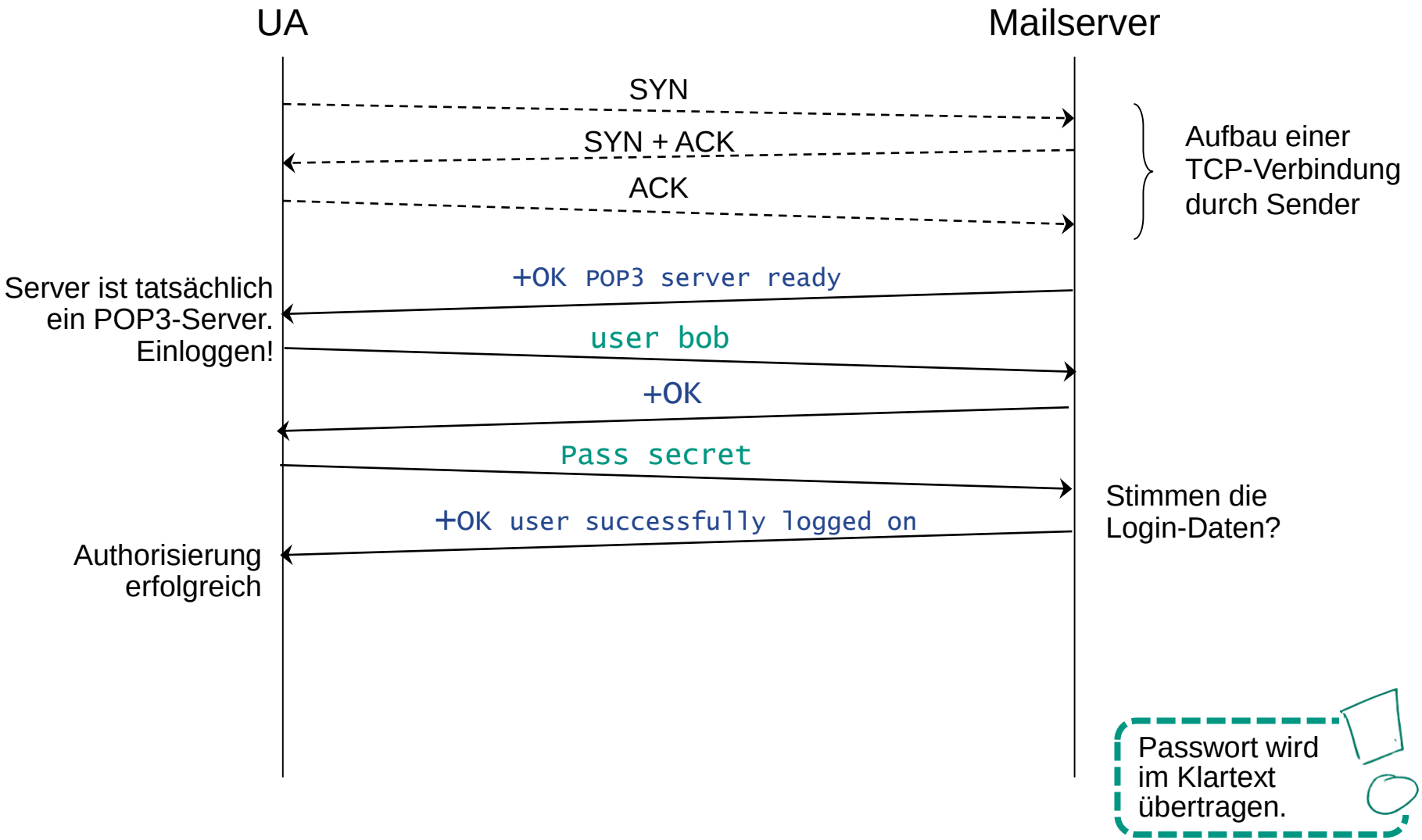
Abfrage des Postfachs

- E-Mail zentral über einen Mailserver abgewickelt
  - Abfrage des Postfachs durch weitere Protokolle wie POP3 oder IMAP
  - Auch Web-basierte Abfrage des Postfachs ist möglich

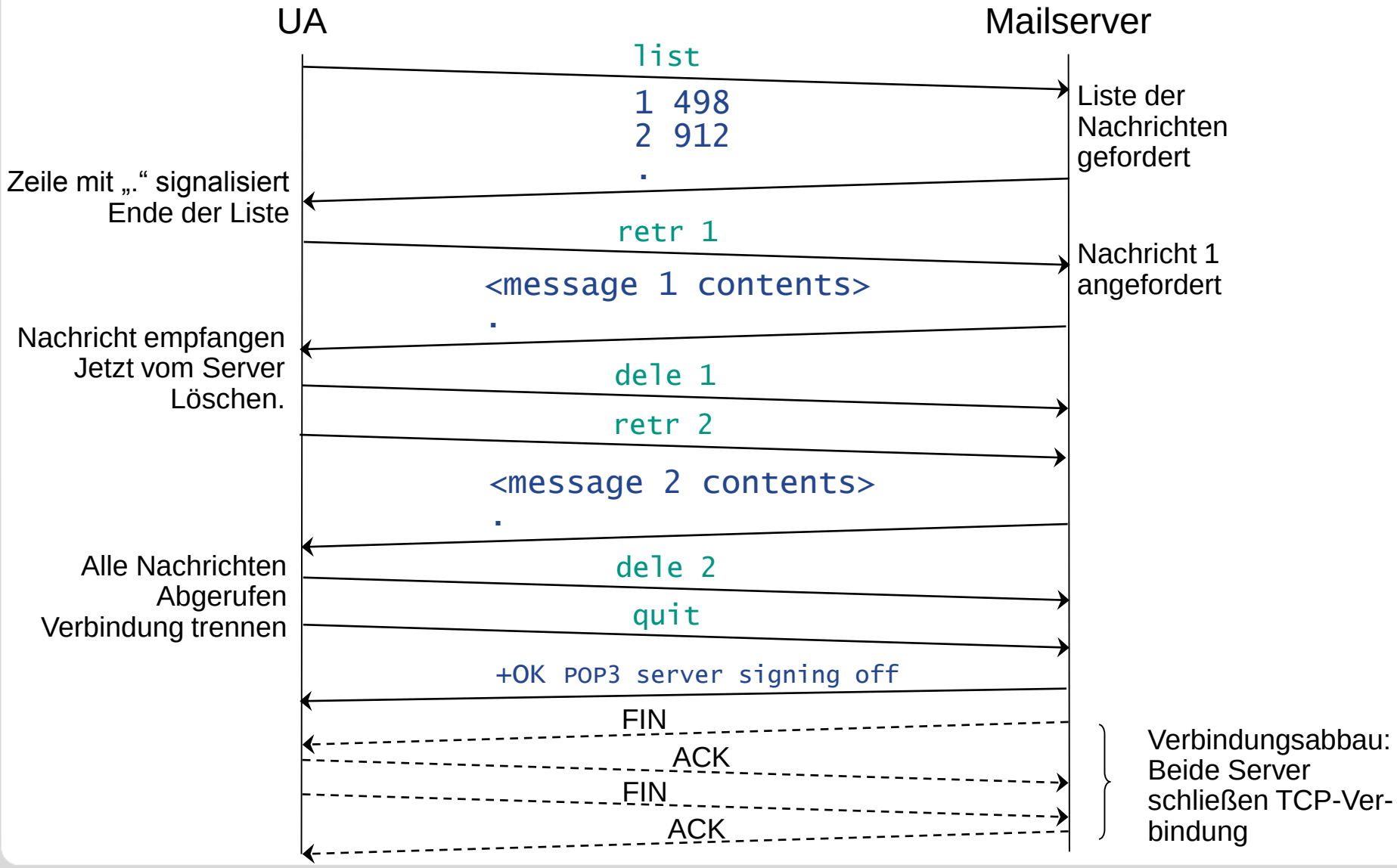
- Mittels POP3 (Post Office Protocol 3) holt der Client die vom Mailserver empfangenen und in der Mailbox gespeicherten Nachrichten ab
  - Einfache Funktionalität
  - Verwaltet Nachrichten im UA
    - Keine Synchronisation zwischen mehreren UAs

- IMAP (Interactive Mail Access Protocol) dient dazu, die Nachrichten zentral auf einem Mailserver zu verwalten
  - IMAP erlaubt erweiterte Kommandos (z.B. Ordnerverwaltung, Filterung)
  - Verwaltet Nachrichten auf dem Mailserver

POP3: Protokoll – Authorization Phase



POP3: Protokoll – Authorization Phase



Web-Mail

- Probleme von E-Mail UAs
  - Installation notwendig
  - Konfiguration erfordert Fachkenntnisse (Server, Ports, …)
  - Support vom Mailbetreiber nicht möglich

- Lösung: Webmail
  - Installation eines Webservers auf dem Mailserver
  - Dieser stellt Inhalte der Mailbox als HTML dar
  - Kann vom Nutzer wie jede andere Website sofort genutzt werden
  - Beispiel am KIT: Outlook Web App

- Nachteile gegenüber herkömmlicher UAs
  - Bindung des Nutzers an Funktionalität des Anbieters
  - Ende-zu-Ende-Verschlüsselung nicht möglich

Pingo

http://pingo.upb.de/

Unsichere E-Mail

- Problem: Fehlende Nutzer-zu-Nutzer-Sicherheit
  - Keine Vertraulichkeit und Integrität von Nachrichten
  - Keine Authentifizierung von Sender und Empfänger

- Angriffe
  - Mitlesen von Nachrichten
  - Manipulation oder Fälschen von Nachrichten

- Geforderte Schutzziele
  - Vertraulichkeit der Nachricht
  - Integrität der Nachricht
  - Authentizität des Senders
  - Authentizität des Empfängers

- Abhilfe: de-facto Standard OpenPGP (→ Kap. 7)

[RFC4880]

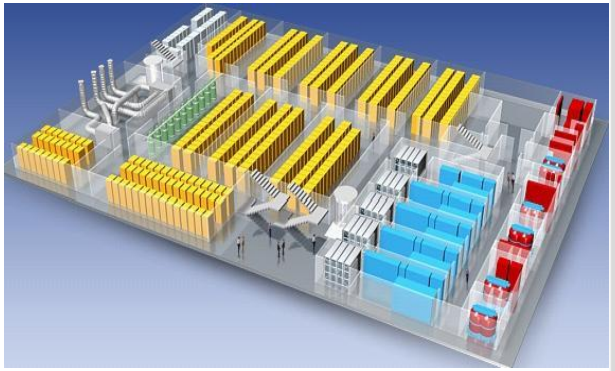
E-Mail-Server

- E-Mail ist asynchroner Nachrichtenaustausch
- Empfangen und Bereithalten der Nachricht für den Empfänger
  - verschiedene Formen möglich
    - von einfachen Dateien bis zur Datenbank
  - hohe Anforderungen an Skalierbarkeit
    - viele Abfragen und quasi pausenlos eingehende E-Mails
    - KIT: 6 Millionen E-Mails/Monat, ca. 3 E-Mails/s
      1&1: 5 Milliarden E-Mails/Monat, ca. 2000 E-Mails/s
  - Mechanismen, um gleichzeitigen Zugriff zu regeln
    - Locking-Verfahren
    - Backup im Betrieb
- E-Mail-Aktion (Eingang, Abrufen, Löschen, Verschieben) in etwa vergleichbar mit Datenbank-Transaktion

Daten sind aus dem Jahr 2011

Beispiel E-Mail: Das Internet ist nicht virtuell

- Rechenzentrum der 1&1 AG in Karlsruhe*
  - 2000m² Grundfläche UG, 1200m² Dachfläche Technik
  - Investitionskosten: 12 Mio. Euro
  - 11 Räume für 660 Racks
    - mehr als 25000 Server
    - 4000 TeraByte Speicherkapazität
  - 5 Elektroversorgungsblöcke (je 2 MW Leistung)
  - über 8 MW Leistungsaufnahme
    - (Stadt Karlsruhe: ca. 361 MW)
  - über 70.000.000 kWh pro Jahr
  - 5 Generatoren mit jeweils 2.4 MW
    - 48.000 Liter Diesel/Tag
  - 8 Kaltwassersätze mit je 700 kW (= 5600 kW)
  - 61 Umluftkühlgeräte mit je 100 kW



[Eins11]

* Stand: 2011

Kapitel 2.5

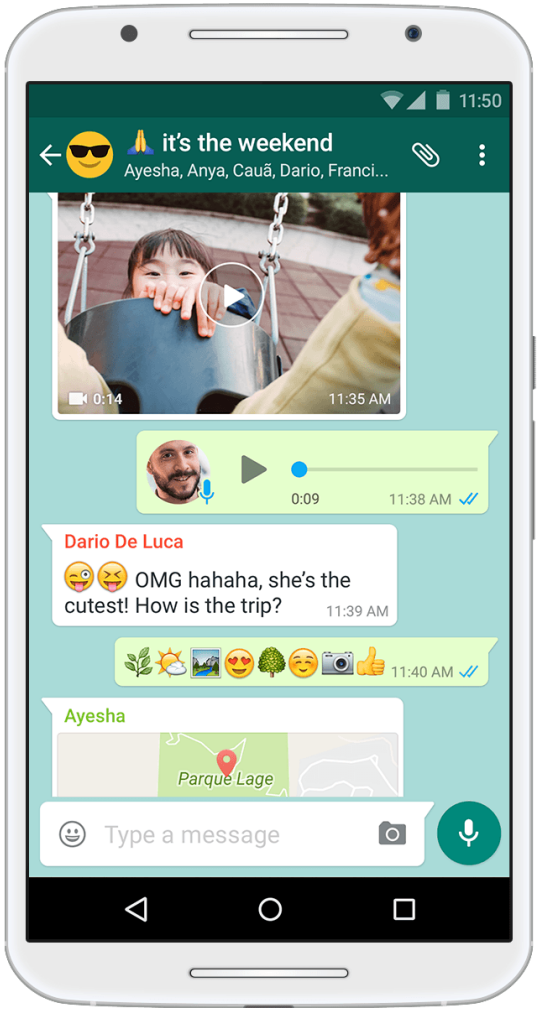
WHATS APP

WhatsApp

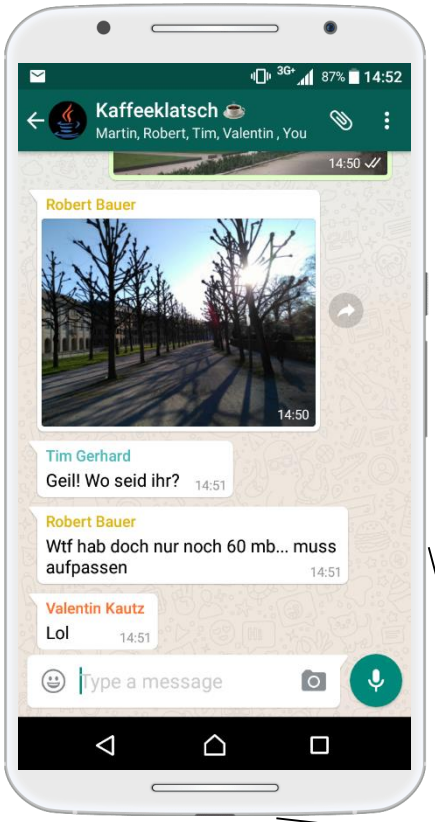
EXAMPLE

Erschien 2008 – derzeit beliebteste App

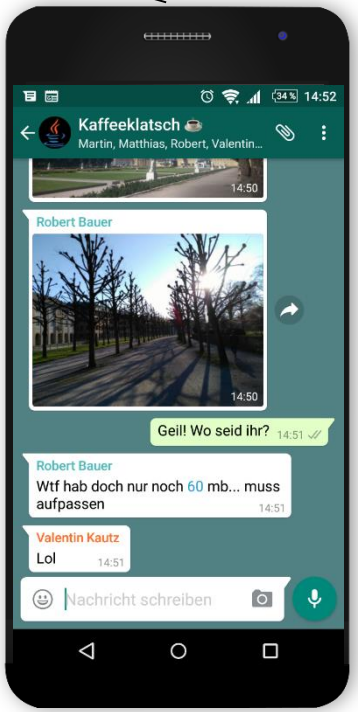
- Text, Fotos, Videos, Dokumente, Lokation, und Videoanrufe
- 1,2 Milliarden Nutzer aus 180 Ländern
- 667 Millionen Nachrichten pro Tag alleine in Deutschland (vs. 39 Millionen bei SMS)
- 100 Millionen Videoanrufe pro Tag (1.100 Anrufe pro Sekunde)
- Android-App Entwicklung: Team von fünf Programmierern (2015)

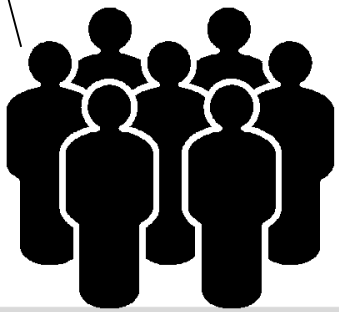


Instant Messaging



Text, Fotos, Videos, …





Interner Aufbau von WhatsApp

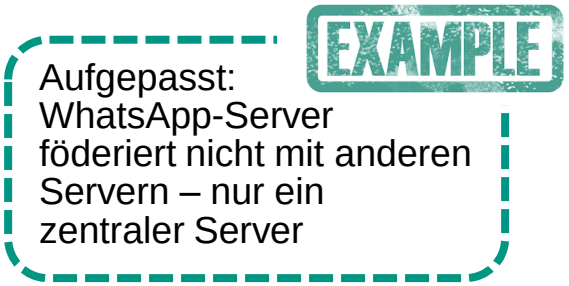
- Proprietäre Anwendung
  - Wird derzeit von Facebook weiterentwickelt
  - Genaue Implementierung / Protokoll unbekannt 😢
  - Aber wohl abgespeckte Version des Anwendungsprotokolls:
    - eXtensible Messaging and Presence Protocol (XMPP)
  - Inoffizieller Name FunXMPP

- XMPP – Echtzeit-XML-Streaming-Protokoll
  - Offener Standard für “near-real-time exchange of data”
    - z.B. zum Versenden von asynchronen Chat-Nachrichten, für Videotelefonie, …
  - Verwendet XML als Nachrichtenformat → Erweiterbar
  - Dezentralisiert, ähnlich wie E-Mail (Domänen / Server)
  - Vorgänger: Jabber (1998, entwickelt durch Jermie Miller)
  - Standardisierung durch: IETF / XMPP Standards Foundation (XSF)

[RFCs 6120-6122, 7622]

XMPP im Überblick

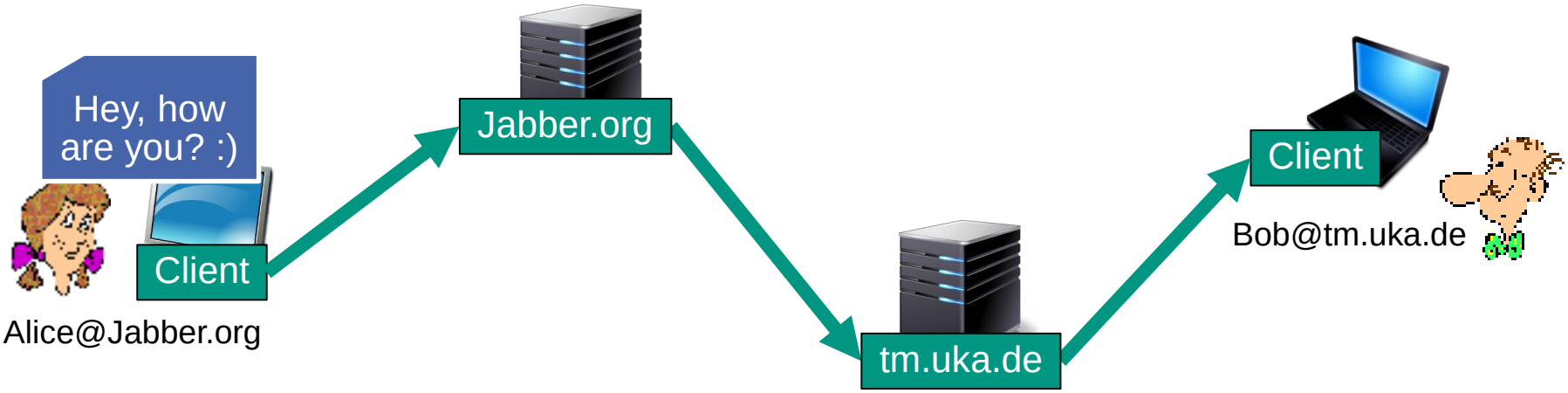
- Clients sind zu ihrem jeweiligen Server verbunden
- Server verbinden sich untereinander zur Nachrichtenübermittlung



- XML-Nachricht
  - enthält beliebigen als XML darstellbaren Inhalt
- Adressformat
  - Nutzer identifiziert durch Server und Username, z.B. Alice@Jabber.org
  - Clients identifiziert pro Nutzer, z.B. Alice@Jabber.org/Laptop

XMPP Beispiel

- Alice sendet XML-Nachricht an Bob



XMPP-Nachricht

- Inhalt sind XML-Dokumente
- Metainformationen
  - Sender und Empfänger
  - Typ
  - …
- Erweiterbares Format
  - Sogenannte “Extensions”
  - Erlauben zusätzliche Funktionalität in Clients und Servern

EXAMPLE

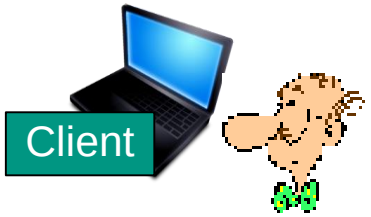
```
<message
    from=‘Alice@Jabber.org/Desktop‘
    id='ktx72v49‘
    to=‘Bob@tm.uka.de‘
    type='chat‘
    xml:lang='en'>

  <body>Hey, how are you? :)</body>

</message>
```

Übermittlung von Alice an Bob

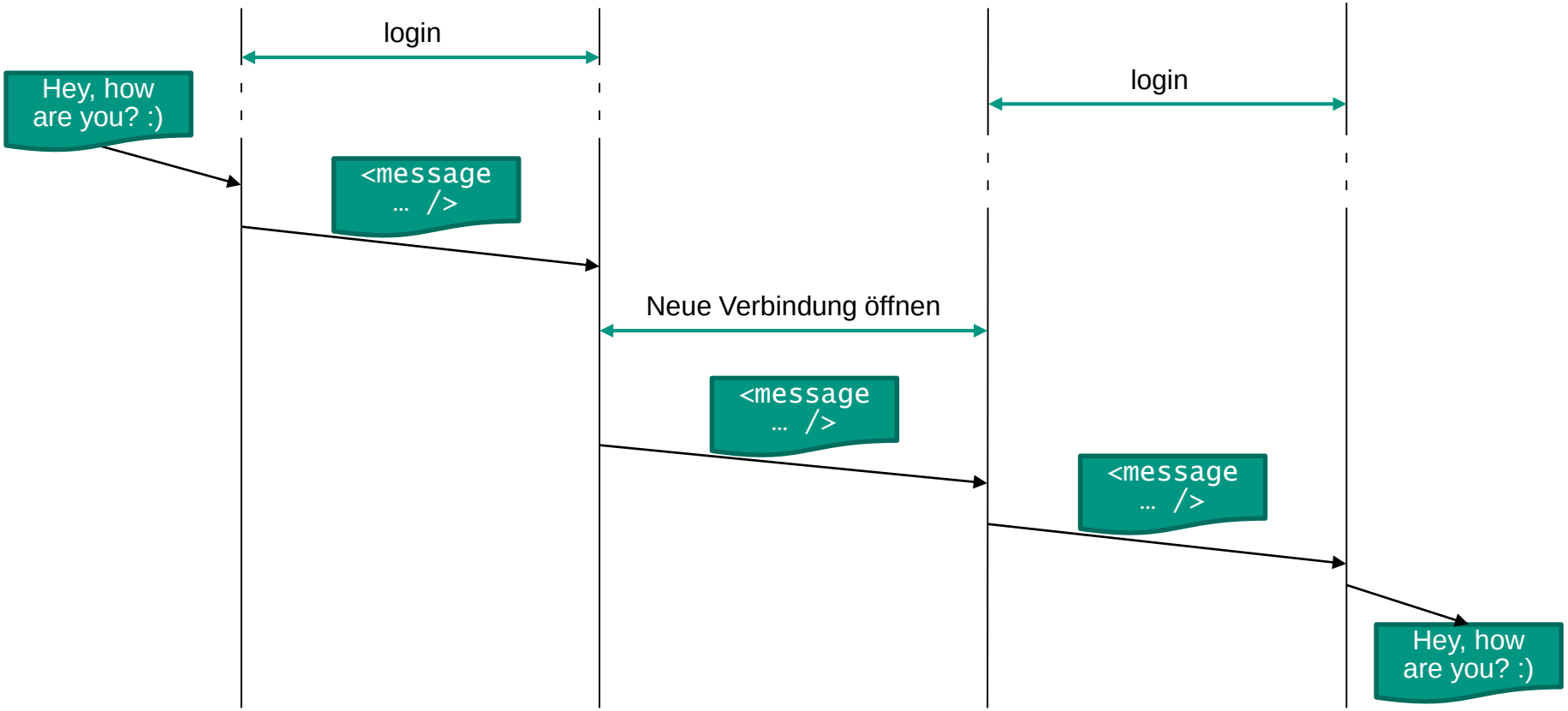
- Alice
  - Falls noch nicht eingeloggt:
    - Baut XML-Stream zu ihrem Server auf
  - Verpackt / codiert Inhalt als XML-Nachricht
  - Überträgt XML-Nachricht per XML-Stream an Ihren Server
- Server von Alice
  - Empfängt XML-Nachricht von Alice
  - Ermittelt Server von Bob
    - Baut ggf. XML-Stream zu Bob‘s Server auf
  - Überträgt XML-Nachricht an Bob
- Server von Bob
  - Empfängt XML-Nachricht von Alice‘s Server
  - Ermittelt Bob‘s Client(s)
    - Falls kein angemeldeter Client, auf Verbindung warten
  - Überträgt XML-Nachricht zu den Clients via XML-Streams
- Bob
  - Empfängt XML-Nachricht
  - Entpackt / decodiert den in der XML-Nachricht enthaltenen Inhalt

XMPP: Nachrichtenübermittlung

Alice@Jabber.org/Desktop



Bob@tm.uka.de/Laptop



Übersicht der Funktionalität in XMPP

- Core – XMPP-Technologien zum XML-Streamen
  - Auf- und Abbau von XML-Streams
  - XML-Stanzas
    - XMPP-Message → eben erklärt, Nachrichtenübertragung
    - XMPP-IQ → Information Queries, z.B. Abrufen der Kontaktliste
    - XMPP-Presence → Austausch von Statusinformationen (Online, Offline, etc.)
- Weitere Teilprotokolle
  - Jingle – SIP-Kompatible Multimedia-Unterstützung für Audio/Video/…
  - Multi-User Chat – Flexible Gruppenchats
  - PubSub – erweiterte Benachrichtigungs- und Präsenz-Funktionalitäten
  - BOSH – XMPP über HTTP
- Zusätzlich: XMPP Extensions
  - Mobile, Service Discovery, Avatare, etc.
  - Müssen allerdings jeweils Clientseitig unterstützt werden
    - Manche auch Serverseitig

Reality Check: XMPP vs. WhatsApp

Offener Standard	vs.	Proprietäre Umsetzung
Fokus auf Erweiterbarkeit	vs.	Keine Erweiterungen
Dezentralisiert / Föderation	vs.	Zentralisiert

- WhatsApp und Sicherheit?
  - Alle Nachrichten gehen über die zentralen WhatsApp-Server
  - Allerdings: Ende-zu-Ende-Verschlüsselung („Signal Protocol“)
  - → Mitlesen von Nachrichten unterwegs nicht möglich!

Kapitel 2.6

DOMAIN NAME SYSTEM (DNS)

Überblick

- Ziel: Verwendung von Namen statt IP-Adressen zur Adressierung von Rechnern (insbesondere Servern)
  - Vergleichbar einem Telefonbuch
  - Namen sind für Menschen leichter zu merken als IP-Adressen
- Beispiel: telematics.tm.kit.edu → 129.13.182.160
- Aufgabe
  - Zuordnung zwischen Name und IP-Adresse
- Funktionalitäten
  - Registrierung von Namen und zugehörigen IP-Adressen
    - Vergabe für „.de“-Domain durch DENIC eG, …
  - Auflösung von Namen in IP-Adressen
    - Verteilte Datenbank mit einer Hierarchie von Name-Servern (DNS-Servern)

[RFC1034/1035 + Updates]

So funktioniert DNS

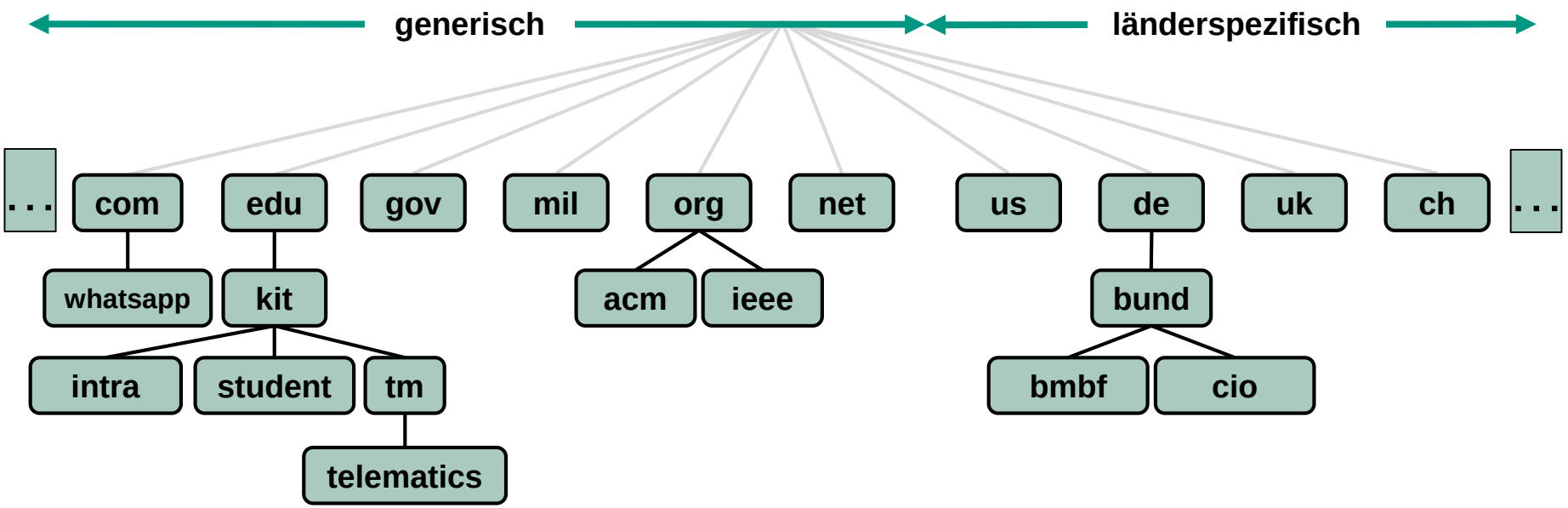
https://www.youtube.com/watch?v=2ZUxoi7YNgs

Exkurs: Bedeutung des DNS heutzutage

„Ausfall der .de-Domain“ am 12. Mai 2010

[Heis10b]

Struktur des Namensraums



- Struktur
  - Hierarchische Struktur: Subdomäne . Domäne . Top-Level-Domain (TLD)
  - Getrennt durch Punkte
- Top-Level-Domains
  - Länder-Domains (.de, .us, .uk, …) nach ISO 3166-1
  - Generische TLDs (.gov, .edu, …, .museum, …, .xxx) festgelegt durch die ICANN (Internet Corporation for Assigned Names and Numbers)

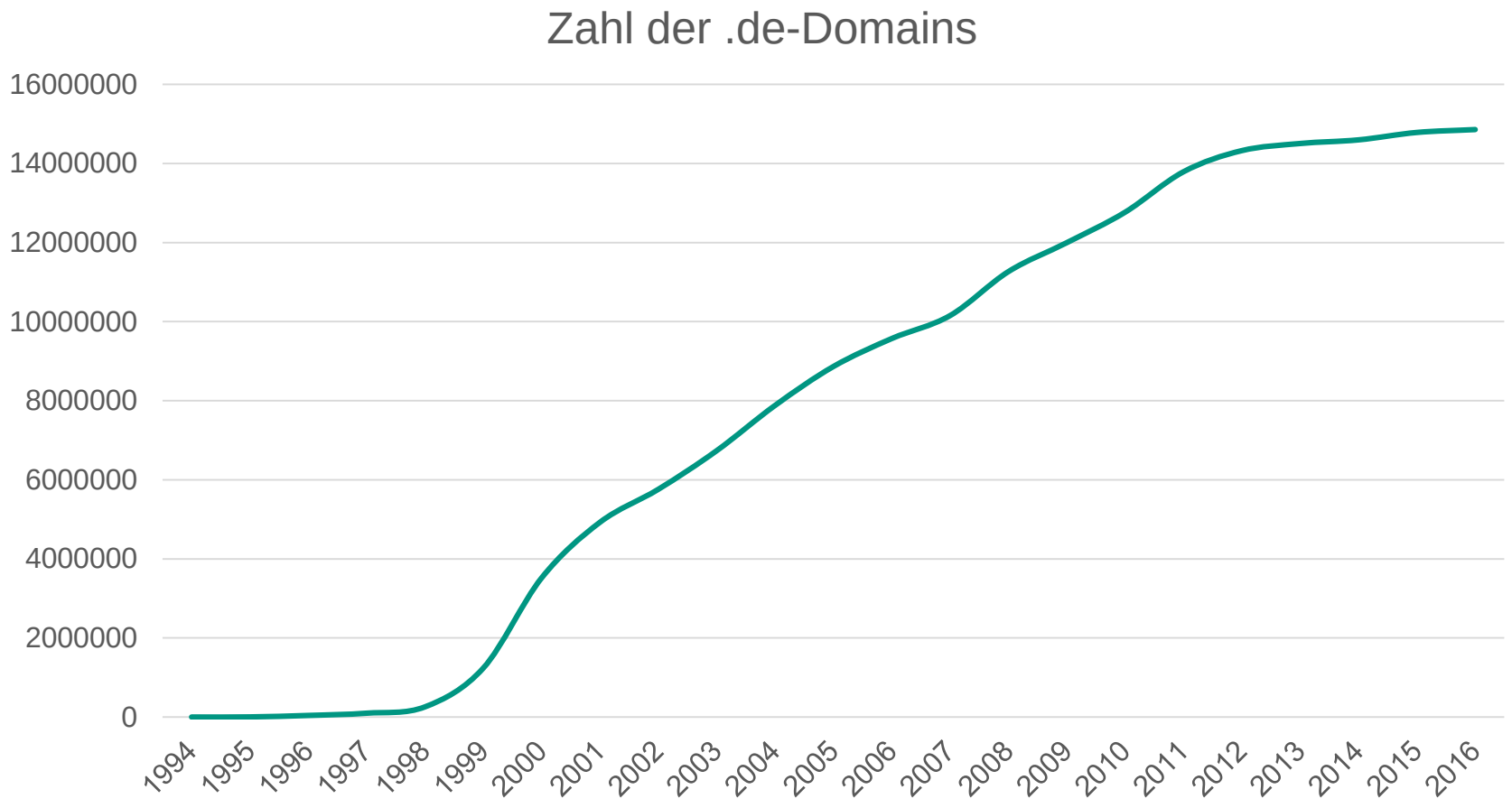
Einführung neuer Top-level Domains (New gTLD)

- ICANN startet Vergabe für neue Top-Level-Domains (New gTLDs)
  - Bewerbungsphase vom 12. Januar 2012 bis 20. April 2012
  - Für Organisationen und Unternehmen
  - Nicht-Lateinische Schriftzeichen
  - Bsp.: .music, .eco, .green, .africa, .berlin, .游戏
- Allerdings hohe Auflagen
  - Bewerbungsgebühr beträgt 185.000 Dollar
    - Spezielle Unterstützung für Bewerber aus Entwicklungsländern
  - Genaue Prüfung der Bewerber
    - Versuch, sich gegen alle Arten von Missbrauch abzusichern
  - Strikte Einhaltung von Markenschutzrechten
- Seit Oktober 2013 schrittweise Einführung neuer Domänen
  - Nach interner Prioritätenliste

[ICANN]

Namensverwaltung der .de-Domäne: DENIC

[DENI16]

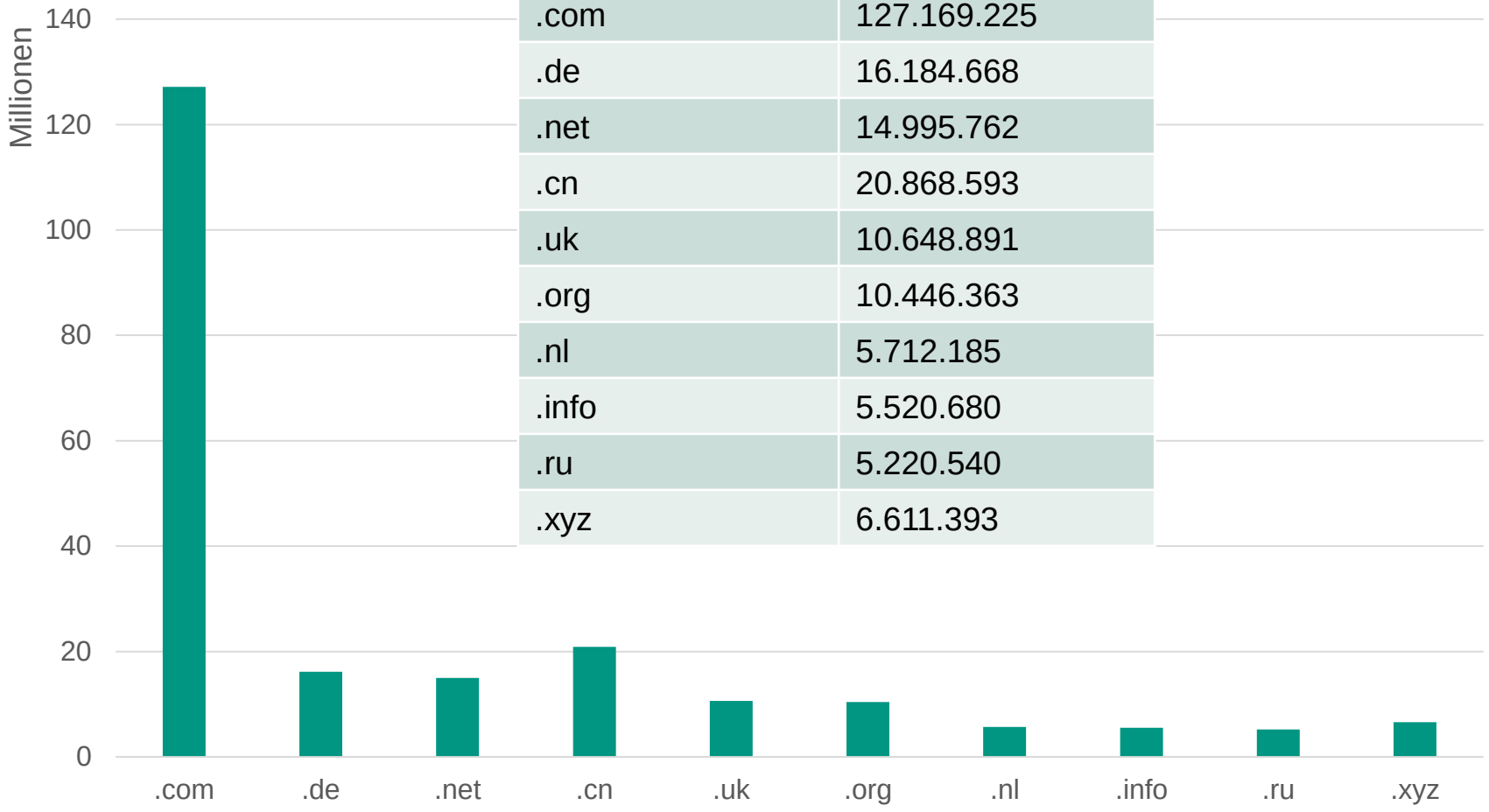


Zahl der .de-Domains am 1. Juli 2015: 15.943.688

Namensverwaltung: Die größten TLDs

[DENI16]

Top Level Domain	Domains weltweit
.com	127.169.225
.de	16.184.668
.net	14.995.762
.cn	20.868.593
.uk	10.648.891
.org	10.446.363
.nl	5.712.185
.info	5.520.680
.ru	5.220.540
.xyz	6.611.393



DNS: ein verteiltes System aus Name-Servern

- Wieso kein zentraler Name-Server?
  - Singuläre Fehlerquelle
  - Verkehrsaufkommen an zentralem Server (weltweit!)
  - Entfernung der zentralen Datenbank
  - Verwaltungsaufwand
  - Weltweite Abhängigkeit
  - → Ein zentraler Ansatz skaliert hier nicht!

- Verteilung von DNS-Servern
  - Kein Server kennt alle Abbildungen von Namen auf IP-Adressen
  - Lokale DNS-Server
    - Im Allg. haben ISPs, große Firmen etc. einen lokalen DNS-Server
      - Name-Server des KIT: dns-int-01.kit.edu, dns-int-02.kit.edu, …
    - Erste Anfrage geht immer zu diesem lokalen Server (Default-Server)
  - Autoritativer DNS-Server
    - Enthält autoritative Abbildungen, d.h. liefert maßgebliche Antworten

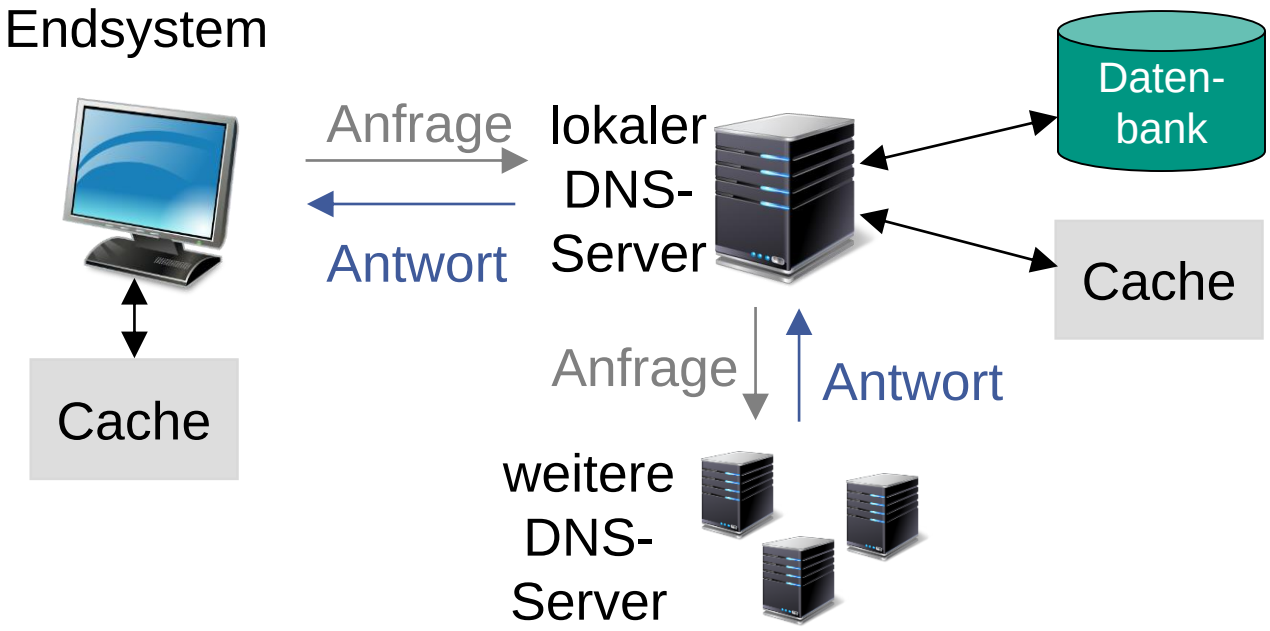
DNS: Aufbau und Eigenschaften

- Verteilte Datenbank mit einer Hierarchie von Name-Servern (DNS-Servern)
  - Client/Server-Modell
  - Server kann Anfrage an andere Server weiterleiten
- Protokoll der Anwendungsschicht
  - Über UDP realisiert – Port 53
  - Wieso nicht TCP?
- Basisdienst im Internet – keine eigentliche Anwendung
  - Komplexität ist am Rande des Netzes lokalisiert
  - → Internet Design Philosophie! (Ende-zu-Ende Paradigma)
- Verwendung
  - Wird von anderen Protokollen der Anwendungsschicht eingesetzt
    - HTTP, SMTP, FTP, …

[RFC1034/1035]

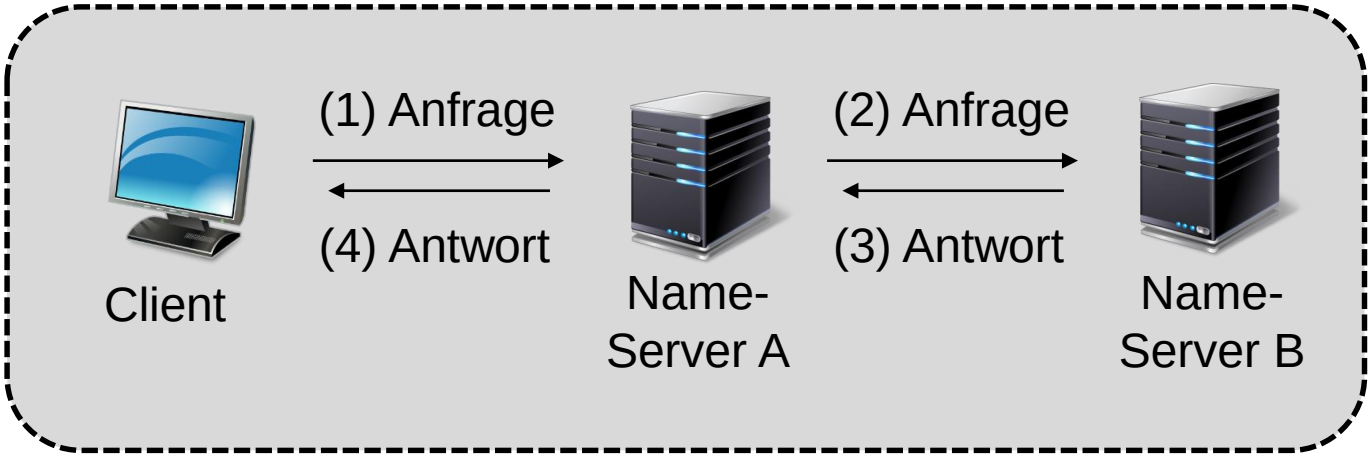
DNS-Anfrage

- Wenn ein Endsystem den gefragten Namen nicht kennt (nicht im eigenen Cache hat), wird lokaler DNS-Server befragt
- Lokaler DNS-Server antwortet entweder
  - aus seiner eigenen Datenbank mit Einträgen (falls autoritativ)
  - aus seinem Cache gespeicherter Antworten auf vorangegangene Anfragen
  - nach Befragung anderer DNS-Server, falls er die Antwort nicht liefern kann



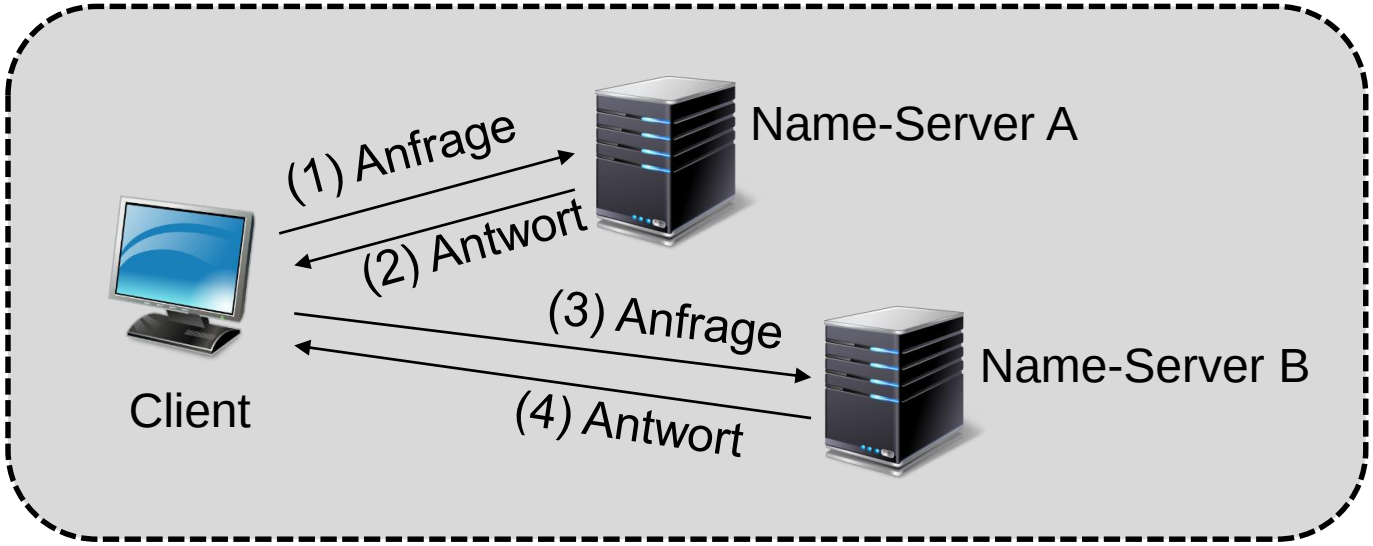
Rekursive Anfrage

- Kennt angefragter Server die Antwort nicht, befragt er weitere Server, bis er die Antwort zurückliefern kann
- Vorteil: Geringer Aufwand für Client, da Arbeit zur vollständigen Namensauflösung dem Server überlassen wird



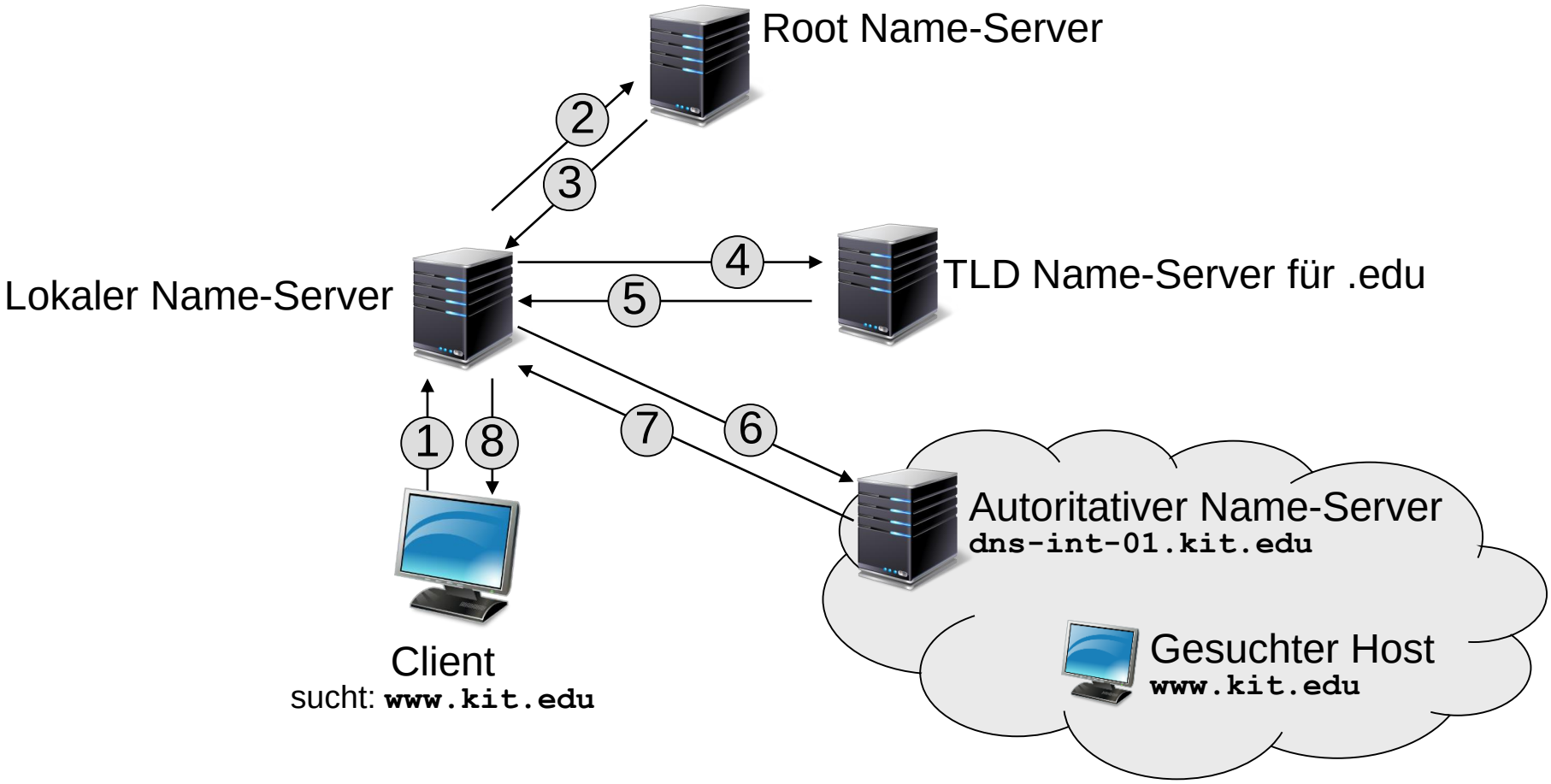
Iterative Anfrage

- Kennt angefragter Server die Antwort nicht, kann er mit Verweis auf anderen Server antworten
- Client befragt dann weitere Server
- Vorteil: Weniger Aufwand für angefragte Name-Server



Typischer Ablauf

- Typischerweise fragt Client seinen lokalen Name-Server rekursiv
- Lokaler Name-Server arbeitet dann normalerweise iterativ



Hierarchie der Name-Server

- Name-Server hierarchisch gegliedert
  - Root-Server
    - Weltweit einige wenige solcher Server (bzw. Server-Cluster)
  - Top-Level Domain (TLD) Server
    - Verantwortlicher Server für die Top-Level Domains
  - Autoritative Name-Server
    - Jeder Host ist bei einem solchen Server registriert
    - Häufig gleichzeitig lokaler Name-Server in „seinem“ Netz
  - Lokale Name-Server
    - z.B. Provider, Universität, Firma etc. betreiben lokalen Name-Server
    - Adresse des lokalen DNS-Server manuell konfiguriert oder per DHCP
- Für jede Hierarchie-Ebene (= Domäne) gilt
  - Die IP-Adresse mindestens eines Root-Servers ist bekannt
  - Kennt Name-Server, die für die nächst tiefere Ebene zuständig sind
  - Jeder Name-Server enthält die Einträge, für die er autoritativ ist
    - Besitzt Autorität zur Namensvergabe innerhalb dieser Domäne
  - Ergebnisse von vorhergehenden Anfragen werden zwischengespeichert
    - Timer legt Dauer der Zwischenspeicherung fest

DNS Root-Server

- IP-Adressen und Lokation der DNS Root-Server unter http://www.root-servers.org
  - IPv4 und bei der Mehrheit auch IPv6-Adressen
  - IP-Adressen sind fix
- Root-Server enthalten nur Einträge für TLDs (Generische und Länder-TLDs)
- Weltweit gibt es 13 Root-Server(-Betreiber)
  - Bezeichnung „A“, …, „M“
  - Bspw. wird der K-Root-Server von RIPE NCC betrieben
  - Root-Server besteht i.d.R. aus mehreren Server-Instanzen
    - insgesamt 632 Server (Stand Oktober 2016)
    - Anycast-Dienst

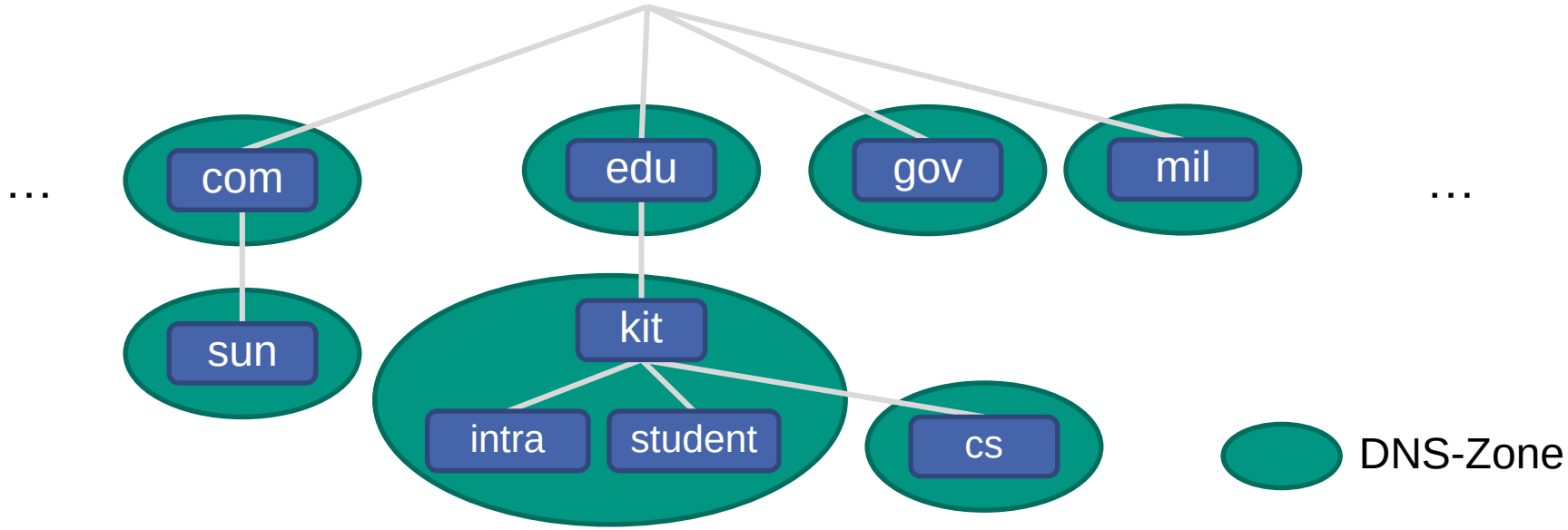
[https://www.root-servers.org] (Stand: April 2017)

Weltweit verteilte Instanzen des K-Root-Servers

[https://www.ripe.net/analyse/dns/k-root/] (Stand: April 2017)

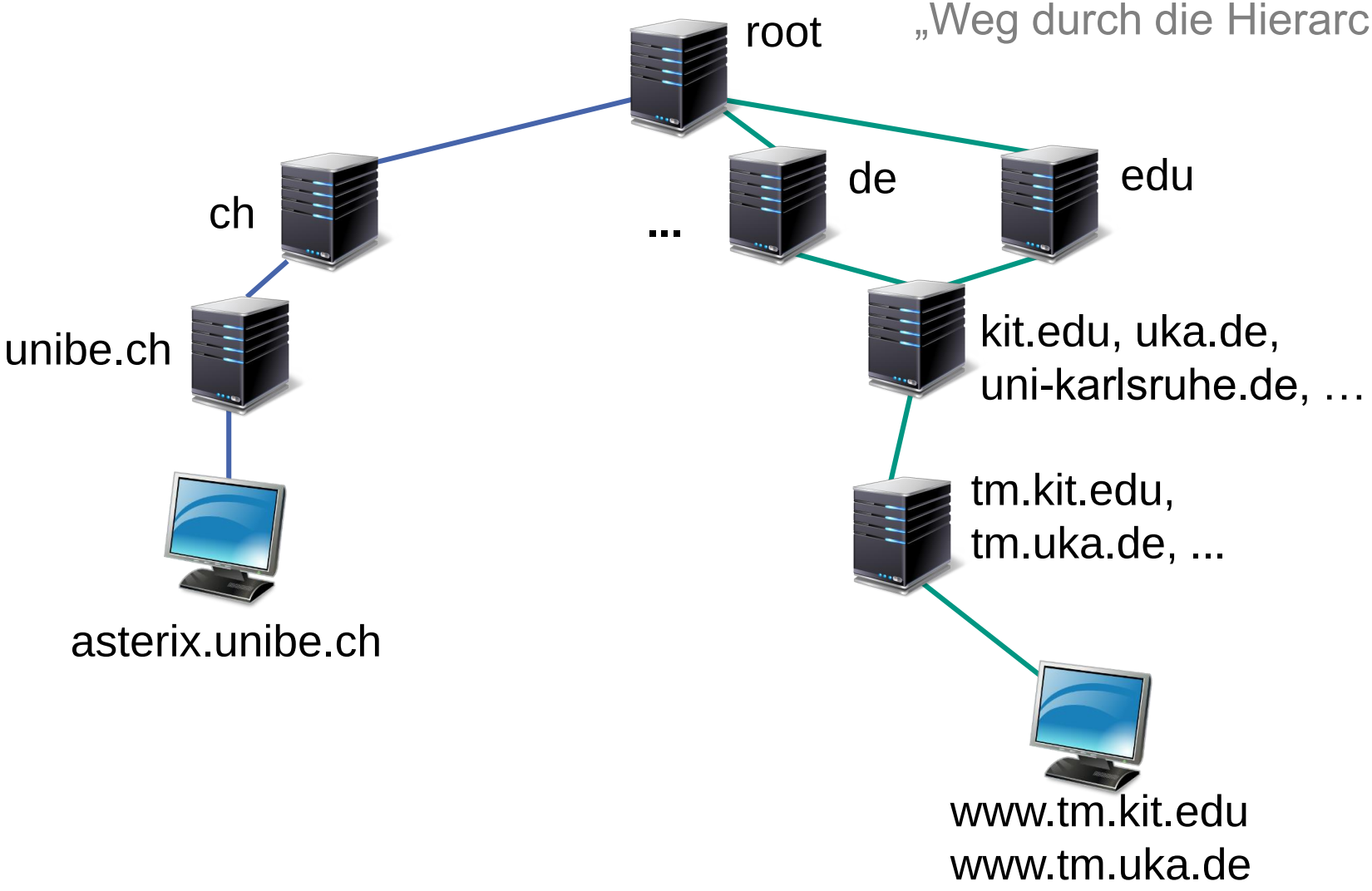
- Von RIPE NCC angebotene Instanzen des K-Root-Servers
  - Verteilte Struktur erlaubt besseren Schutz gegen Ausfallsicherheit und kürzere Antwortzeiten durch näher gelegene Server
  - Globale Knoten sind uneingeschränkt global erreichbar
  - Lokale Knoten sind nur für die Kunden der angebundenen ISPs vorgesehen
    - Z.B. Kunden, die mit DE-CIX in Frankfurt ein sog. Peering haben

DNS-Zonen



- Zonen
  - Repräsentieren den Teil des Domänenbaums für den ein Name-Server zuständig ist
  - Kann mehrere Subdomänen enthalten
- Zonendatei beschreibt eine Zone
  - Enthält die Ressource Records der Zone
  - Genau ein SOA RR (Start of Authority): Konfiguration der Zone (Gültigkeit, Version)
- Zonentransfer: Spiegelung der Zonendatei auf weitere Nameserver (Redundanz, Performance)

Nameserver-Hierarchie: Beispiel



Weitere Dienste von DNS

- Host Alias
  - Endsysteme mit komplizierten Namen können über einen oder mehrere Alias-Namen verfügen
    - Alias-Namen sind typischerweise einfacher zu merken
    - DNS kann für einen Alias-Namen den „eigentlichen“ Namen liefern (kanonischer Name)
- Mail Server Alias
  - E-Mail-Adressen sollten so gestaltet sein, dass sie leicht zu merken sind, z.B. praesident@kit.edu
  - Der Host-Name des Mail-Servers ist in vielen Fällen komplexer gestaltet
    - z.B. für praesident@kit.edu der scc-mailin-01.scc.kit.edu
- Lastverteilung
  - Bei redundant ausgelegten Servern (z.B. Server-Cluster) kann eine Menge von IP-Adressen mit einem Namen assoziiert sein
    - DNS-Server antwortet mit gesamter Menge der zugehörigen IP-Adressen, verändert aber die Reihenfolge, in der diese genannt werden
    - Client wählt einen beliebigen Server aus

DNS-Einträge: Resource Records (RR)

- DNS ordnet Domänen zu Einträgen zu (sog. Resource Records)

- Varianten von Resource Records

Typ	Beschreibung	Wert
A bzw. AAAA (Address)	Abbildung Name auf IPv4/IPv6-Adresse	IP-Adresse
MX (Mail Exchange)	E-Mail-Server einer Domäne	IP-Adresse
NS (Nameserver)	Name-Server einer Domäne	Hostname
CNAME (Canonical Name)	„Alias“-Namen für Rechner/Domänen	Domain
PTR (Pointer)	Abbildung IP-Adresse auf Name	Domain

- Weitere Ressource Records-Typen definiert in RFC 1035
  - http://www.iana.org/assignments/dns-parameters

[RFC1035]

Beispiel: DNS-Anfrage mit dig

```
bash-4.0$ dig  kit.edu MX

; <<>> DiG 9.3.4-P1.2 <<>> kit.edu MX
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51346
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;kit.edu.                       IN      MX

;; ANSWER SECTION:
kit.edu.                3600    IN      MX      10 scc-mailin-01.scc.kit.edu.
kit.edu.                3600    IN      MX      10 scc-mailin-02.scc.kit.edu.
kit.edu.                3600    IN      MX      10 scc-mailin-03.scc.kit.edu.

...

bash-4.0$ dig scc-mailin-01.scc.kit.edu A

; <<>> DiG 9.3.4-P1.2 <<>> scc-mailin-01.scc.kit.edu A
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 37589
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;scc-mailin-01.scc.kit.edu.     IN      A

;; ANSWER SECTION:
scc-mailin-01.scc.kit.edu. 3600 IN      A       129.13.185.193
```

Windows:

```
>nslookup kit.edu
```

Linux:

```
$ dig kit.edu
```

DNS – Sicherheit

- Problem: Alter DNS-Standard hat keine Sicherheitsmechanismen
  - Zentraler Basisdienst des Internets
  - Fälschen von DNS-Einträgen möglich
    - Erlaubt das Umleiten von Datenverkehr

- DNS Security Extension (DNSSEC)
  - Ziel: Gewährleistung von Authentizität und Integrität von DNS-Transaktionen

- Signierte DNS-Antworten mittels asymmetrischer Kryptografie
  - Besitzer einer Zone signiert jeden einzelnen Record mit geheimem Schlüssel
  - DNS-Clients prüfen mittels öffentlichem Schlüssel

Pingo

http://pingo.upb.de/

Kapitel 2.7

VIDEOSTREAMING UND CONTENT DELIVERY NETWORKS

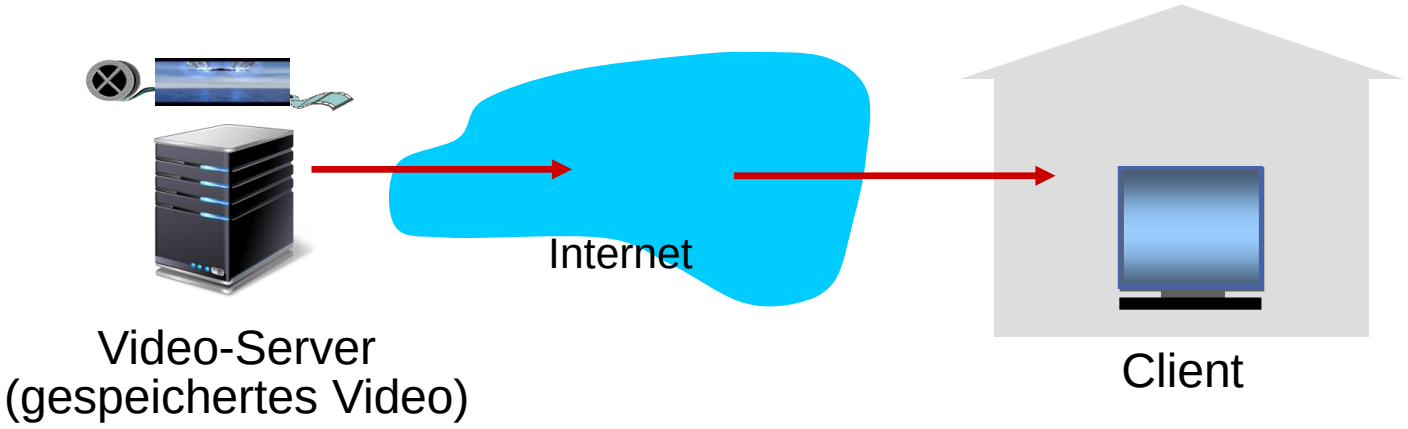
Video Streaming

- Bei Internet-Nutzern äußerst beliebt

- Aus Netzperspektive
  - Videos erfordern hohe Datenrate bei hoher Datenqualität
  - Video mit 2 Mbit/s und Dauer von 67 Minuten hat Volumen von 1 Gigabyte
  - 4K Streaming: Ultra-HD-Qualität
    - Ultra-HD1: Film von 90 Minuten hat ca. 75 Gigabyte
    - 25 Mbit/s als Untergrenze von Netflix angegeben

- Komprimierung von Videos
  - Verschiedene Stufen möglich
  - Video in unterschiedlichen Qualitäten bereitstellen
    - Z.B. per hochauflösendem Fernseher oder per Smartphone ansehbar

Einfaches Szenario für Videostreaming



HTTP-Streaming

- Video auf HTTP-Server in Videodatei gespeichert

- Abruf eines Videos
  - DNS-Auflösung
  - Aufbau TCP-Verbindung zu Server
  - Videodatei wird in HTTP-Response gesendet
    - So schnell wie Netz es erlaubt
  - Client schreibt Daten in Anwendungspuffer
  - Ab gewissem Pufferfüllstand wird Abspielen gestartet

- Einschränkungen
  - Alle Clients erhalten Video gleicher Auflösung

Dynamic, Adaptive Streaming over HTTP (DASH)

- Server
  - Videodatei in Stücke (Chunks) geteilt
  - Jeder Chunk in mehreren Qualitäten verfügbar
    - Verschiedene Datenraten
  - Manifest-Datei beinhaltet URLs und Infos zu Chunks

- Client
  - Nutzt Manifest, um Informationen über Chunks zu erhalten
    - Wählt Chunk mit größter möglicher Bitrate
      - Misst hierzu aktuell mögliche Bitrate
    - Kann verschiedene Bitraten zu verschiedenen Zeiten anfragen
  - Aufgaben beim Client
    - Wann wird der nächste Chunk angefordert?
    - Welche Bitrate soll dabei gewählt werden?
    - Von wo wird Chunk angefragt?
      - Server sollte nahe am Client sein

Content Distribution

- Herausforderung: Content zu hunderttausenden Nutzern bringen

- Option 1: Einziger, großer “Mega-Server”
  - Gegebenenfalls langer Weg zu einzelnen Clients
  - Netzwerk-Überlastung am Server
  - Single-Point of Failure

Skaliert nicht!

Content Distribution

- Option 2: Nutzung von Content Distribution Networks (CDNs)
  - Content auf geographisch verteilte Server kopieren

  - Privates CDN
    - Gehört z.B. Content Provider selbst
    - Beispiel: CDN von Google
      - Verteilt u.a. Videos (YouTube)

  - Third-Party CDN
    - Verteilt Content für mehrere Anbieter
    - Beispiel: Akamai, Limelight

CDN-Strategien

- Enter Deep
  - Viele, kleine Cluster in Zugangsnetzen der ISPs
  - Nahe beim Nutzer: kürzere Antwortzeiten
  - Viele Standorte: hoher Wartungs- und Verteilungsaufwand

EXAMPLE

Akamei

- Ca. 1700 Standorte weltweit

- Bring Home
  - Wenige, große Cluster in wichtigen IXPs
  - Wenige Standorte: geringerer Wartungs- und Verteilungsaufwand
  - Weiter vom Nutzer weg: höhere Antwortzeiten

EXAMPLE

Limelight

- Weniger als 100 Standorte weltweit

Beispiel: Google

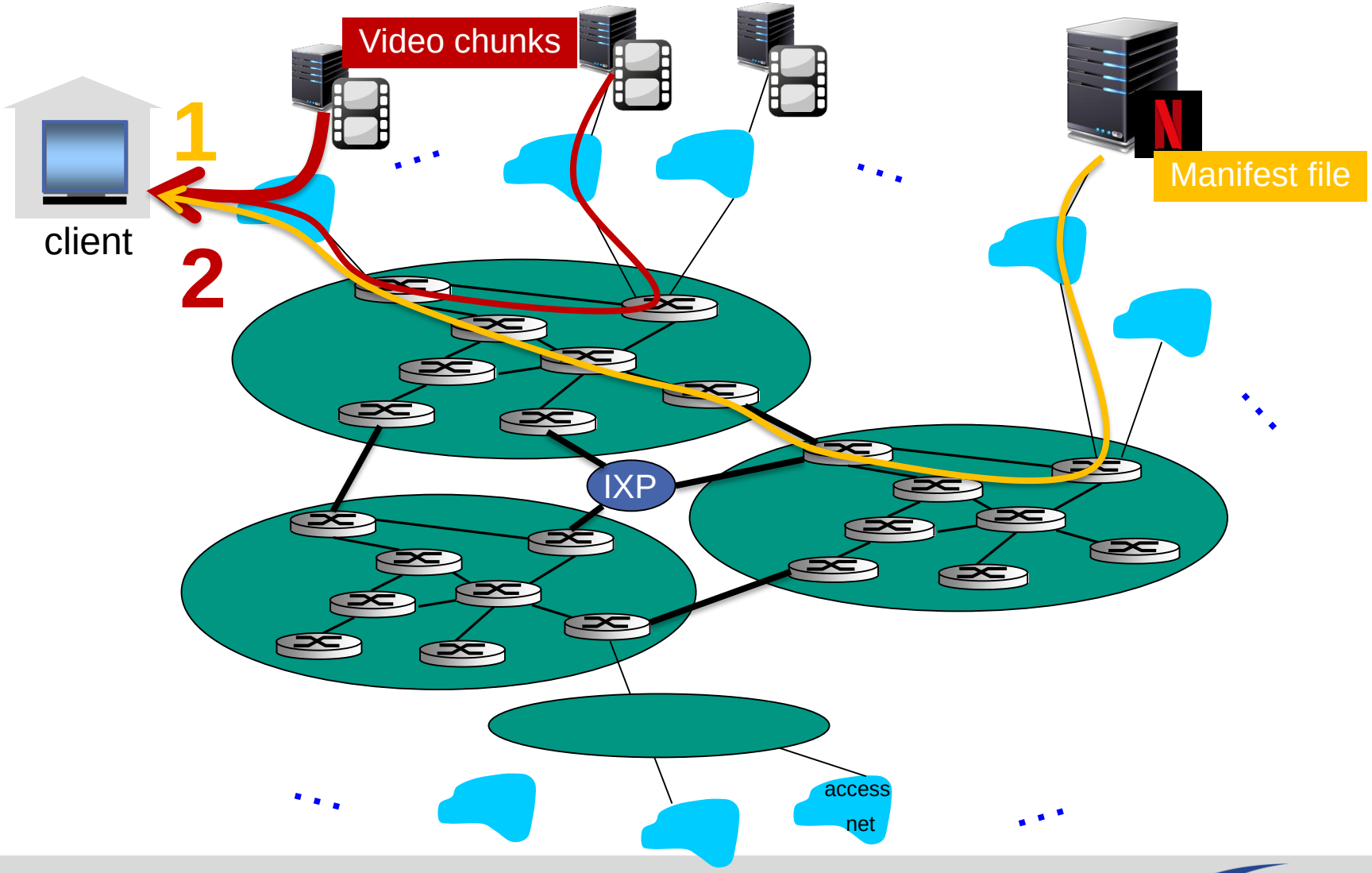
EXAMPLE

Infrastruktur von Google

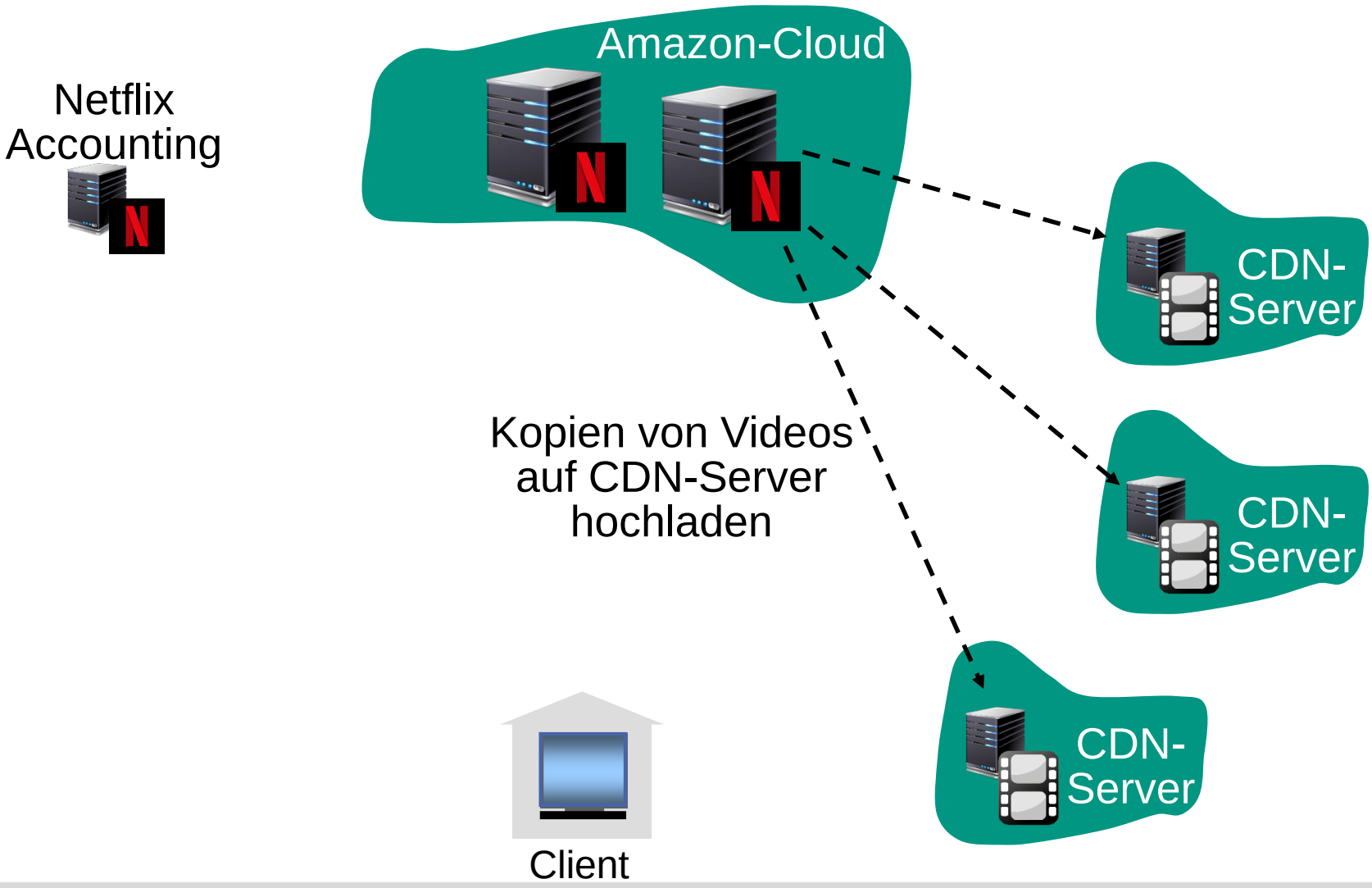
- 14 Mega-Datenzentren (8 Nordamerika, 4 Europa, 2 Asien, je ca. 100.000 Server)
  - dynamische Inhalte
- Ca. 50 Cluster in IXPs (je 100-500 Server)
  - statische Inhalte (u.a. YouTube Videos)
- Mehrere hundert Enter-Deep Cluster bei Access ISPs (einige zehn Server, ein Rack)
  - statische Inhalte, TCP Splitting

- Typischer Ablauf
  - Anfrage über lokalen ISP zu nahem Enter-Deep-Cache; Weiterleiten über Google-Netz an eines der Mega-Datenzentren
  - Video könnte von Bring-Home-Cache kommen, Teile der Webseite von nahem Enter-Deep-Cache und Werbungen von Datenzentren

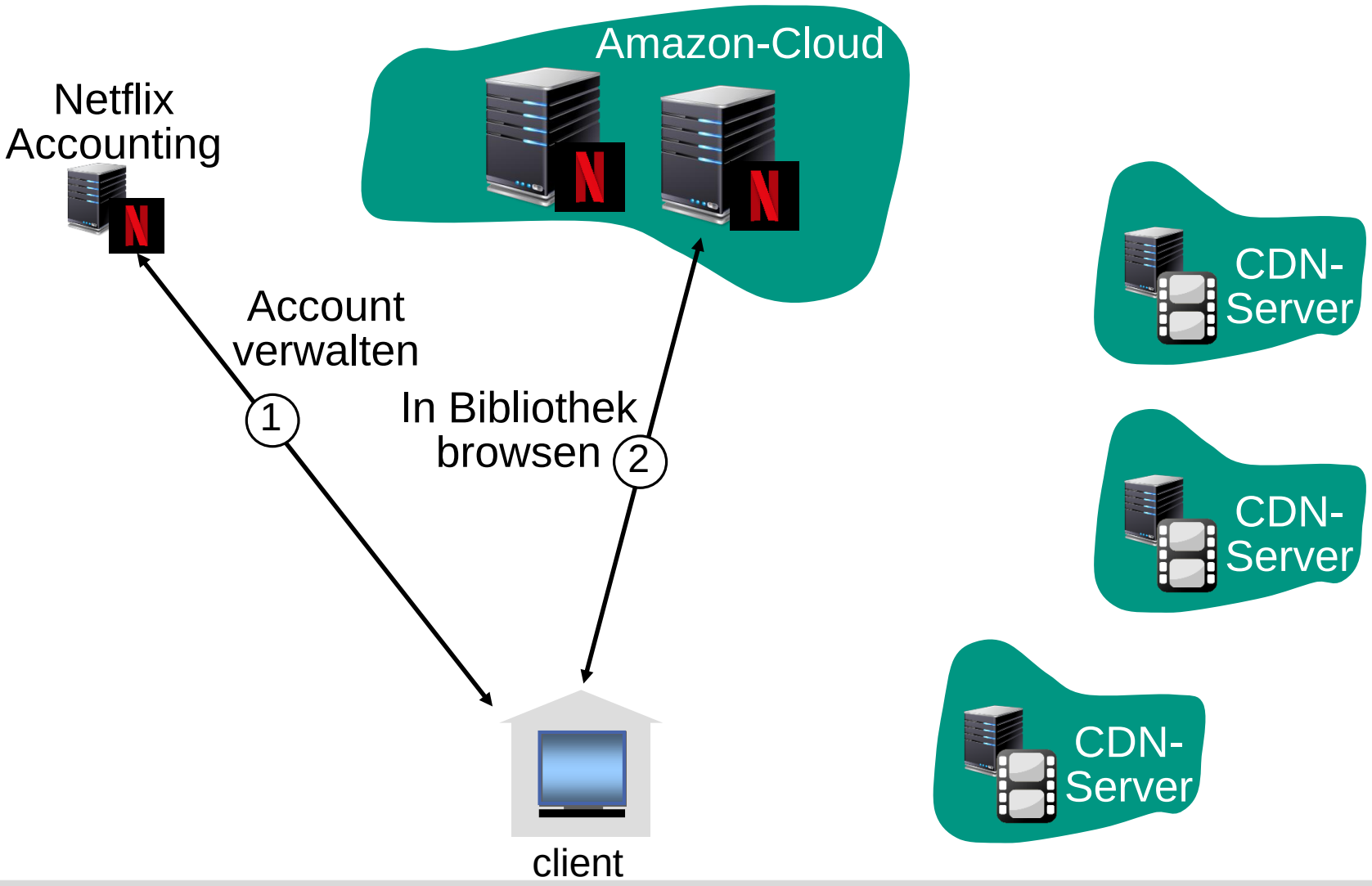
Beispiel: Enter-Deep-CDN und DASH



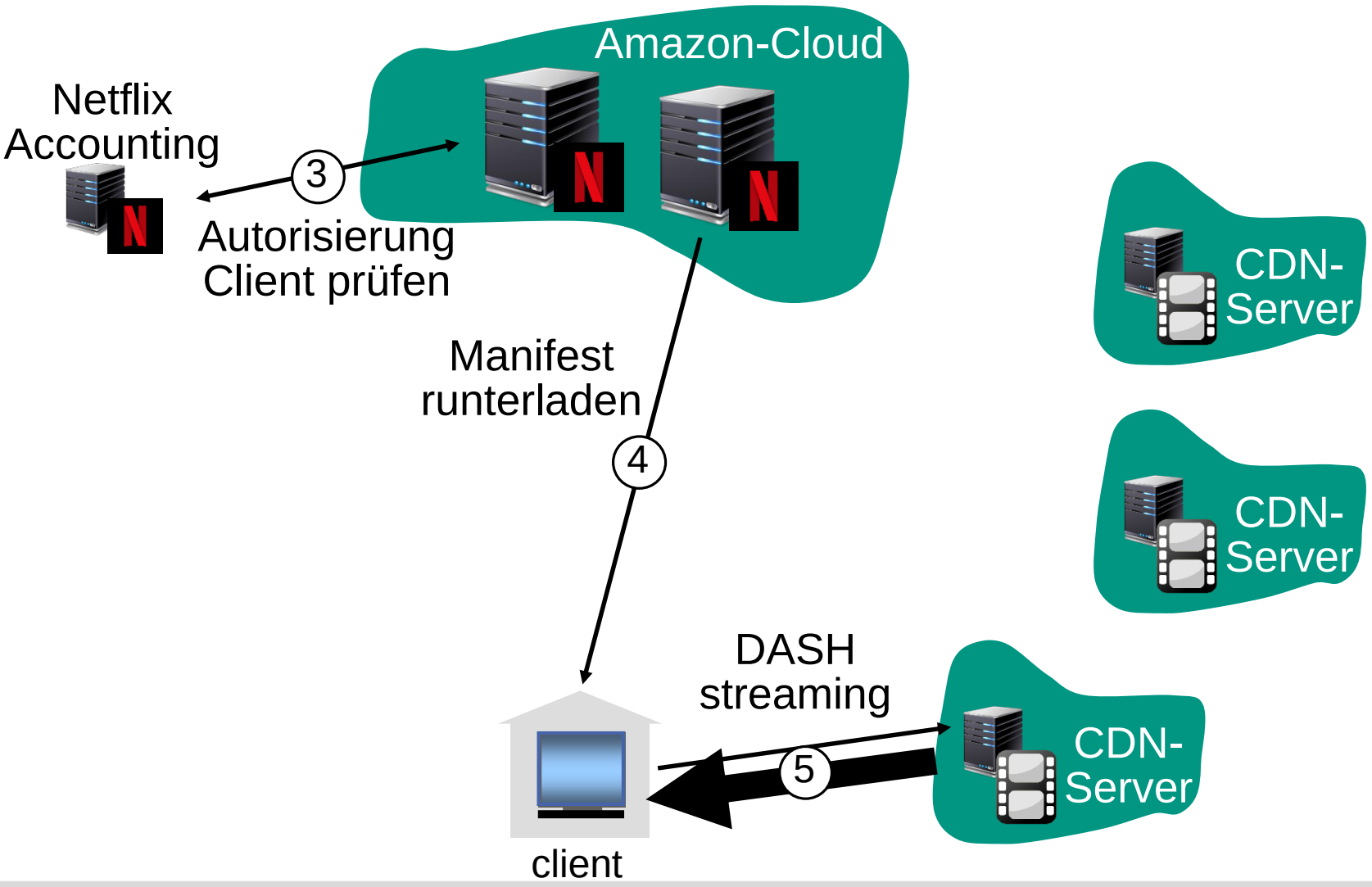
Beispiel: Netflix



Beispiel: Netflix



Beispiel: Netflix

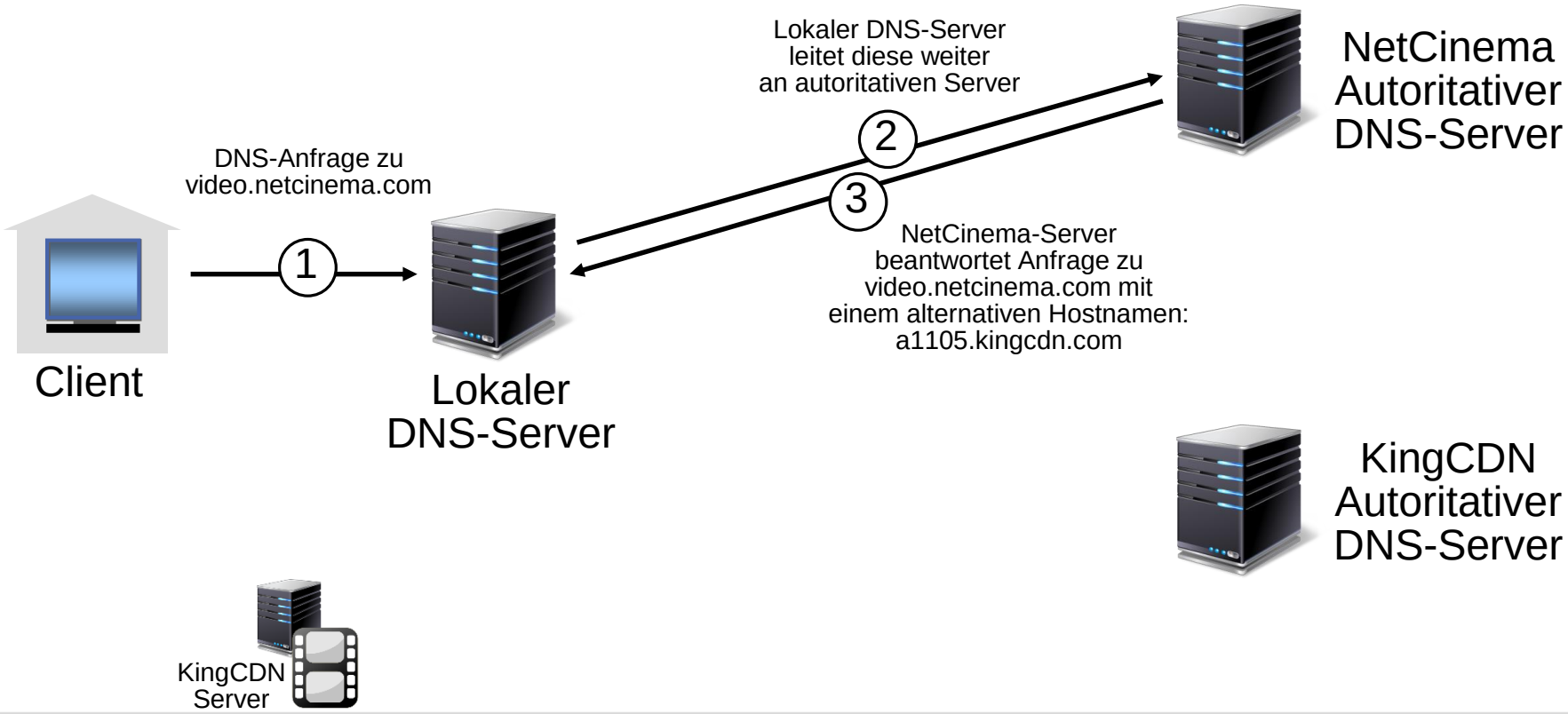


CDN mit “DNS-Manipulation”

- Client muss nahegelegenen CDN-Server finden
  - Manifest-URLs an Client-IP anpassen
    - oder
  - DNS-Manipulation

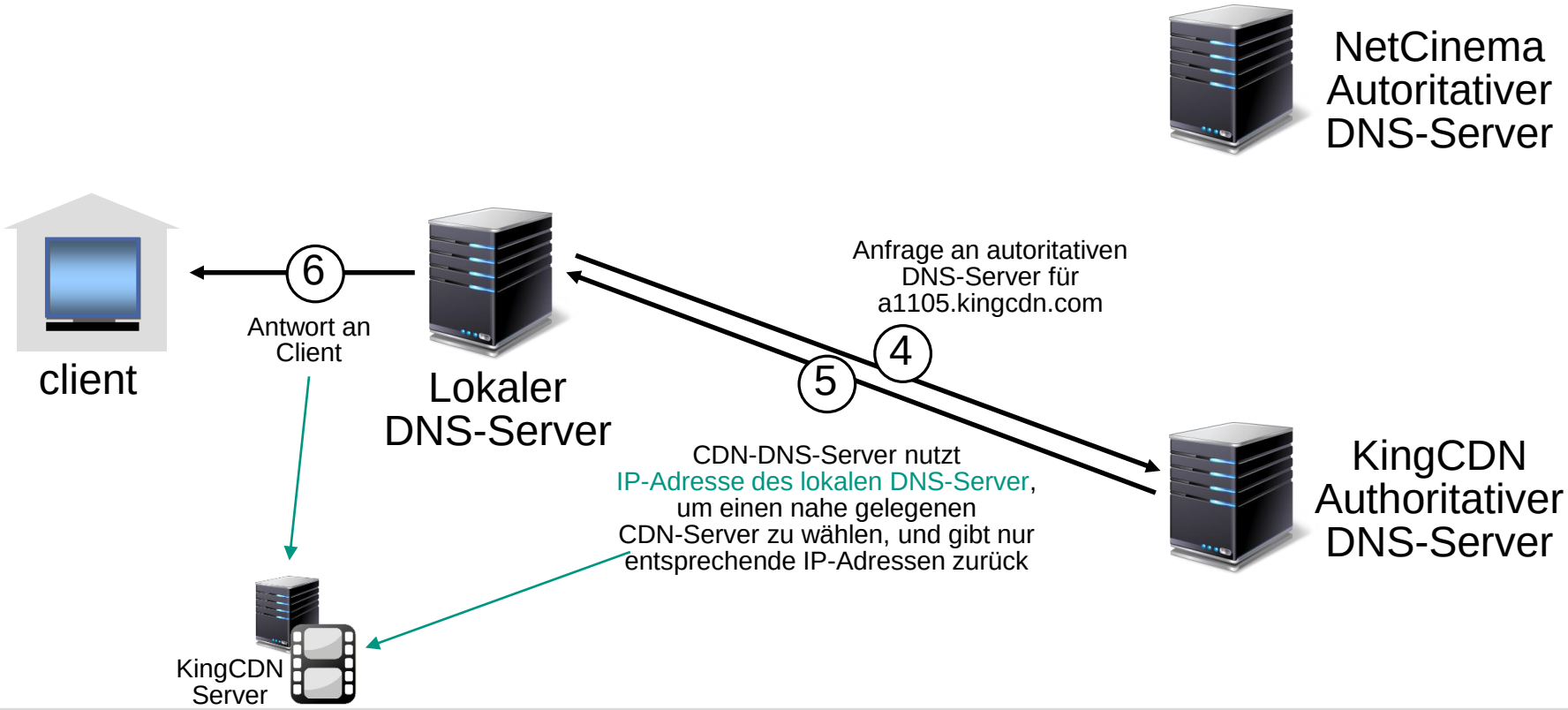
Funktionsweise CDN

- Client kennt Adresse eines Chunks: http://video.netcinema.com/6Y7B23V
  - video.netcinema.com muss noch aufgelöst werden
- Ziel: Umleiten der Anfrage auf lokalen CDN-Server



Funktionsweise CDN

- Client kennt Adresse eines Chunks: http://video.netcinema.com/6Y7B23V
  - video.netcinema.com muss noch aufgelöst werden
- Ziel: Umleiten der Anfrage auf lokalen CDN-Server



Pingo

http://pingo.upb.de/

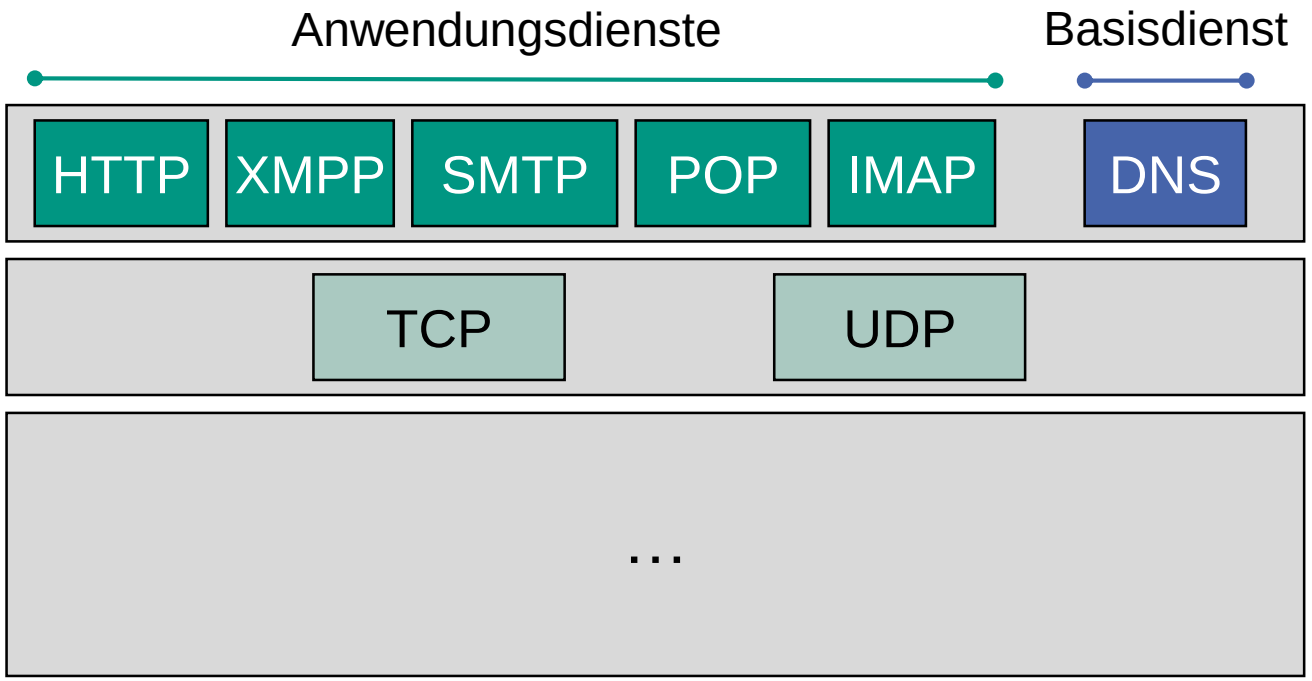
ZUSAMMENFASSUNG …

Verteilte Anwendungen

- Grundlagen zu verteilten Anwendungen
  - Sockets
  - Paradigmen (Client-Server vs. Peer-to-Peer)
  - Verzögerungen
- Einführung in wichtige Anwendungen bzw. Anwendungsprotokolle
  - Web und HTTP
  - Electronic Mail
  - XMPP (im Kontext WhatsApp)
  - Domain Name System (DNS)
  - Videostreaming und Content Delivery Networks (CDNs)

Zusammenfassung

- Anwendungsprotokolle: oberste Schicht im Internet-Protokollstack
  - Nutzung der Protokolle der Transportschicht (TCP, UDP)
  - Anwendungsdienste: HTTP, XMPP …
  - Basisdienst: DNS



1) Was versteht man unter einer verteilten Anwendung? Wo werden sie ausgeführt?
2) Was ist ein Prozess? Was ist eine Nachricht?
3) Wozu benötigt man Sockets?
4) Was ist das Client-Server-Modell? Was ist die Alternative dazu?
5) Wieso entstehen Verzögerungen?
6) Was ist ein URL, wozu dient er und wie ist er aufgebaut?
7) Wie läuft üblicherweise der Abruf einer WWW-Seite über HTTP ab?
8) Welche Methoden unterstützt HTTP, und wozu?
9) Welche Protokolle spielen beim Abruf von E-Mails aus Mailboxen die Hauptrolle und worin bestehen deren wichtigste Unterschiede?
10) Auf welchem Protokoll basiert das WhatsApp-Protokoll?
11) Welche Komponenten gibt es in XMPP?
12) Wie serialisiert XMPP Nachrichten?
13) Wie unterscheidet sich WhatsApp von XMPP?

14) Erläutern Sie die Aufgaben des Domain Name Systems und dessen Aufbau
15) Wie läuft eine Namensauflösung ab und welche unterschiedlichen Anfragetypen können hierbei vorkommen?
16) Beschreiben Sie, worin eine DNS-Antwort besteht und geben Sie Beispiele, was sie enthalten kann
17) Was ist eine DNS-Zone?
18) Wozu nutzt man CDNs?
19) Was ist DASH?

Literatur

- [Arte10] arte, Internet und Geopolitik, Mit offenen Karten, April 2010
- [Bit11] Bittorrent Inc., Pressemitteilung, 1. Januar 2011 (online verfügbar)
- [Cisco10] Cisco Inc., Cisco Visual Networking Index: Forecast and Methodology, Juni 2010 (online verfügbar)
- [Eins11] 1&1 Internet AG, http://www.1und1.de/UnternehmenRechenzentren (2011)
- [Heise09] Heise Online, Philips stellt Net TV vor, Februar 2009, http://www.heise.de/newsticker/meldung/Philips-stellt-Net-TV-vor-201269.html
- [Karr08] D. Karrenberg, DNS Root Name Servers - Frequently Asked Questions, Internet Society Member Briefing, Februar 2008, http://www.isoc.org/briefings/020/
- [Labo10] C. Labovitz, S. Lekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, Internet Inter-Domain Traffic, August 2010, ACM SIGCOMM 2010, Neu-Deli, Indien
- [Radi09] radicati Group, Mai 2009, http://www.radicati.com/?p=3237
- [Skype06] S. A. Baset und H. Schulzrinne, An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol, April 2006, IEEE INFOCOM 2006, Barcelona, Spanien
- [Spie06] K.-P. Kerbusk, M. Rosenbach und Th. Schulz, Alles wächst zusammen, Spiegel 07/2006 (online verfügbar)
- [Veri10] VeriSign, The Domain Name Industry Brief, Februar 2010, http://www.verisign.com/domain-name-services