Kapitel 3: Anwendungsschicht

MNM TEAM

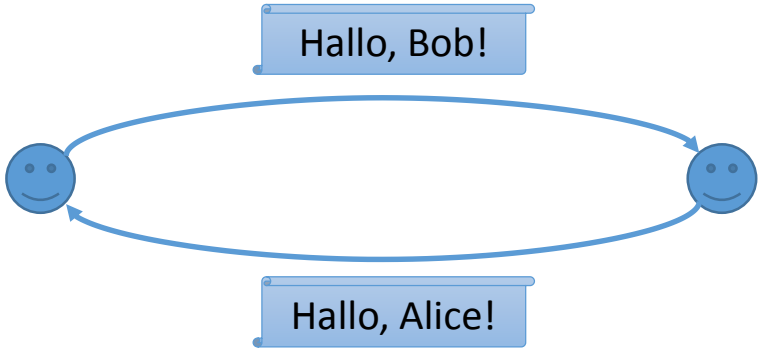
Inhalt von Kapitel 3

Namen bzw. Adressen kommen in allen Schichten und Protokollen zum Einsatz.

1. Namen und IP-Adressen
2. DNS – Domain Name System
3. Client-Server-Anwendungen

Chat-Beispiel: Adressierung

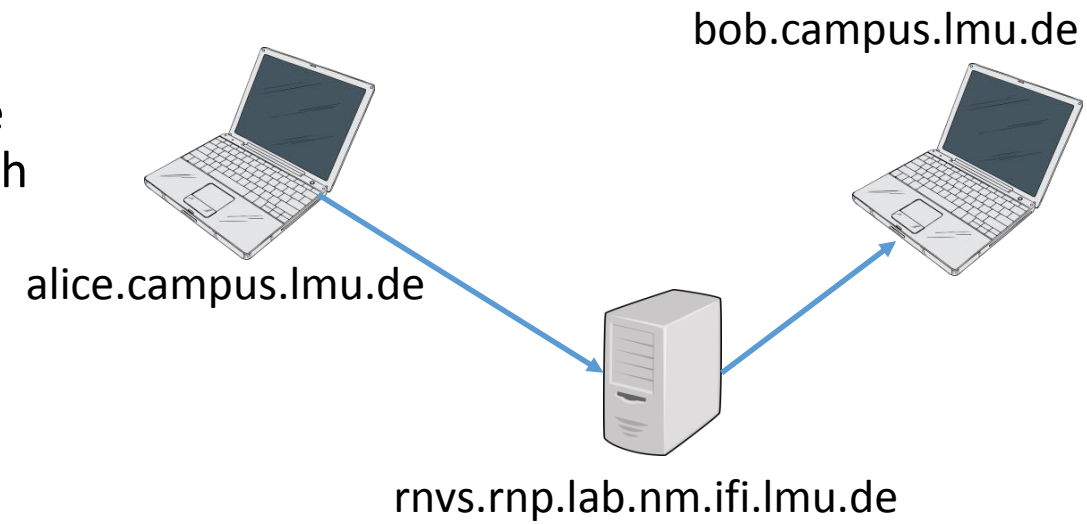
Adressierung des **Empfängers** einer Nachricht anhand von *mnemonischen* **Benutzernamens**



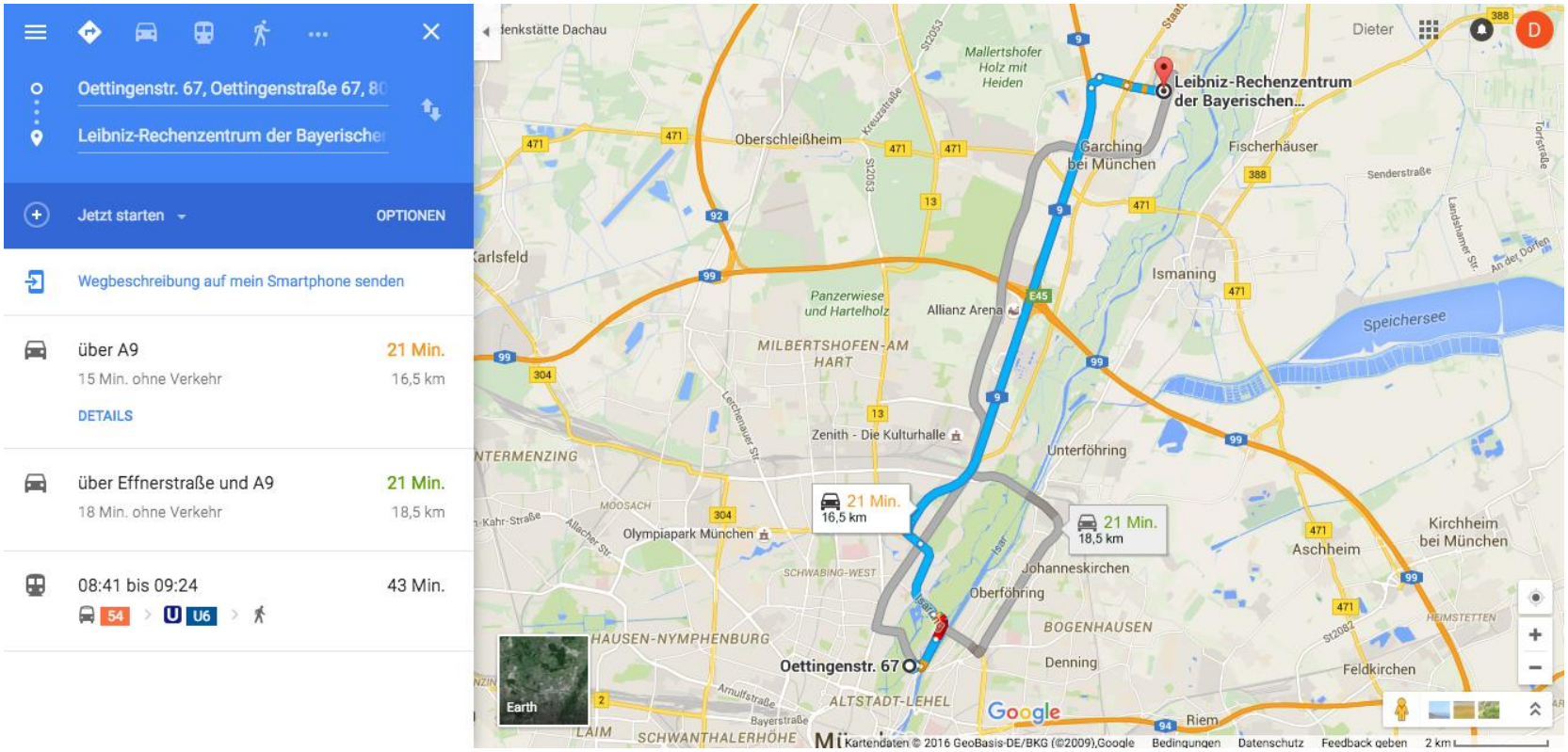
Adressierung der Hosts anhand eines mnemonischen **Hostnamens**

Aber: Hostnamen geben keine Informationen darüber, wo sich ein Host im Internet befindet.

→ IP-Adressen



Namen und Adressen



Kapitel 3.1
Grundkonzepte zu Namen und Adressen

Motivation/Grundproblem

- **WH**: In einem verteilten System wird mittels Austausch von Nachrichten kommuniziert.
- **Grundproblem**:
  Für die Kommunikation ist es notwendig, dass die *Adressen* der Kommunikationspartner bekannt sind bzw. diese *adressiert* werden können
- Zusätzlich:
  - Aus *Adressen* müssen *Pfade* bzw. *Wege* ableitbar sein
  - Zwischenstationen sollten Nachrichten mit gegebener Adressierung so weiterreichen, dass diese möglichst effektiv an ihr Ziel gelangen.

Begriffsklärung

- **Adressen** dienen der Identifizierung von Netzkomponenten oder Diensten
- Adressen müssen von von Maschinen (Computern) effizient verarbeitet werden können
- (Mnemonische) **Namen** sind solche, die Menschen sich gut merken können
- *Namens-* oder *Adressraum* ist eine Menge von eindeutigen Namen bzw. Adressen, die uns in einem bestimmen Format zur Kommunikation in einem Kontext zur Verfügung stehen

Eigenschaften

- (Mnemonische) Namen
  - dienen der Bequemlichkeit menschlicher Nutzer
  - mehrere Inkarnationen logisch identischer Objekte
  - Objekten können ortsunabhängig Namen behalten
  - Bedeutung oft kontextabhängig
  - strukturiert oder flach
- Adressen
  - dienen der effizienten maschinellen Verarbeitung
  - werden für Routing/Wegewahl verwendet
  - strukturiert oder flach

Adressbildung

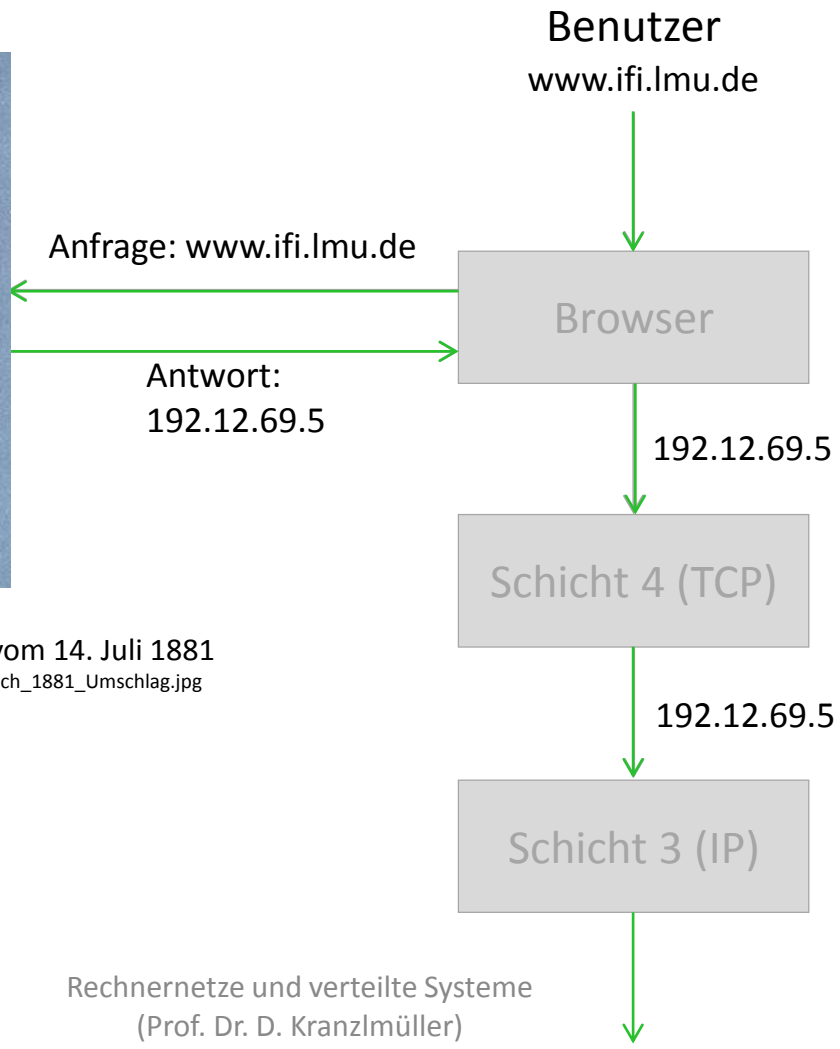
- Struktur
  - uniform global (durchlaufende Nummerierung) (Bsp.: Ethernet-Adressen)
  - hierarchisch (Bsp.: Telefonnummern)
- Umfang des Adressraums
  - groß genug: ermöglicht permanente Zuordnung, erfordert große Header
  - Zu klein: Mehrfachverwendung erfordert dynamische Vergabestrategie (Bsp.: DHCP)
- Codierung
  - variable Namensfelder: erweiterbar
  - feste Namensfelder: effizient/einfach zu implementieren

Lokalisierung von Objekten

- **Namensauflösung**: Adressfindung der Objekte, für die ein Name steht
- Zwei Ansätze:
  - Namen werden durch separaten Dienst z.B. mit Hilfe von Zuordnungstabellen abgebildet
  - Adresse ist aus der Namensstruktur ableitbar: <Prozessname>:=<Netz>.<Subnetz>.<Host>.<ID>
- Im Internet:
  Namensauflösung via DNS ➔ Kapitel 2.3

Beispiel: Namensauflösung Web
(stark vereinfacht)

Umschlagseite des Berliner Telefonbuchs vom 14. Juli 1881
https://commons.wikimedia.org/wiki/File:Berliner_Telefonbuch_1881_Umschlag.jpg



Kapitel 3.2
IP-Adressen

Adressen im Internet

Einordnung

- Internet Protokoll (IP) ist zentrales Protokoll des Internets
- **IP-Adressen** sind Bestandteil des Internet Protokolls.
- IP spezifiziert 2 Protokollversionen
  - IPv4 (32 bit)
  - IPv6 (128 bit)

Grundidee und Vergabe (1/2)

- Jeder Host bekommt eindeutige IP-Adresse
  - Strikt genommen bekommen nicht Hosts, sondern Netzschnittstellen IP-Adressen.
  - Ein Host hat oft mehrere Netzschnittstellen hat (z.B. bei Routern der Fall)
  - Im Heimnetz: Hinter einer Netzschnittstelle verbergen sich mehrere Hosts (➔ NAT).

- Jeder Host (mit IP-Adresse) kann jederzeit an jeden anderen Host (mit IP-Adresse) ein IP-Paket verschicken

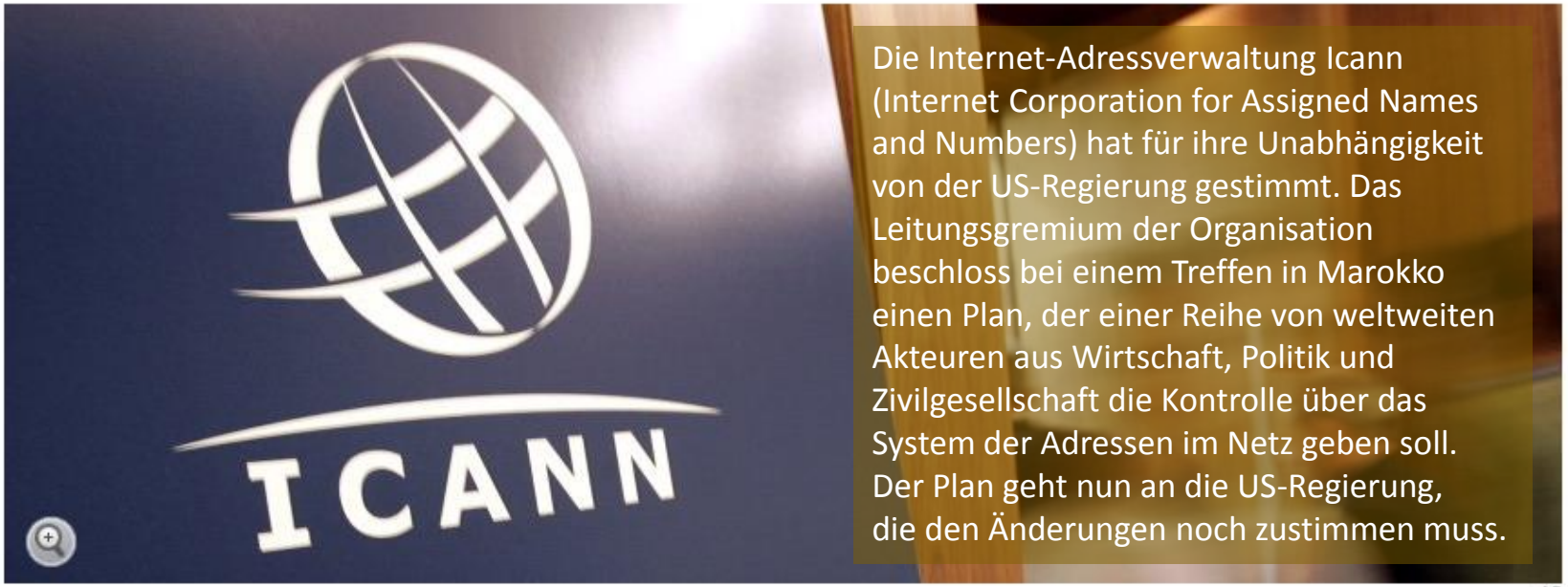
Grundidee und Vergabe

- IPv4-Adressen werden hierarchisch verwendet.
  - IPv4-Adressen bestehen aus Netzteil (Netz ID), der das Netz adressiert, und Hostteil (Host ID), der den Host adressiert
- Einzelne IP-Adressen haben Sonderbedeutungen.

- Internationale Vergabe durch die IANA (Internet Assigned Numbers Authority)
  - Delegiert an nationale Organisationen.
  - Abteilung der ICANN (Internet Corporation for Assigned Names and Numbers)
  - Buchhalter für die Registrierungen

Politik und ICANN

Internet-Adressen: Icann sagt sich von US-Abhängigkeit los



Die Internet-Adressverwaltung Icann (Internet Corporation for Assigned Names and Numbers) hat für ihre Unabhängigkeit von der US-Regierung gestimmt. Das Leitungsgremium der Organisation beschloss bei einem Treffen in Marokko einen Plan, der einer Reihe von weltweiten Akteuren aus Wirtschaft, Politik und Zivilgesellschaft die Kontrolle über das System der Adressen im Netz geben soll. Der Plan geht nun an die US-Regierung, die den Änderungen noch zustimmen muss.

AP

Icann-Logo

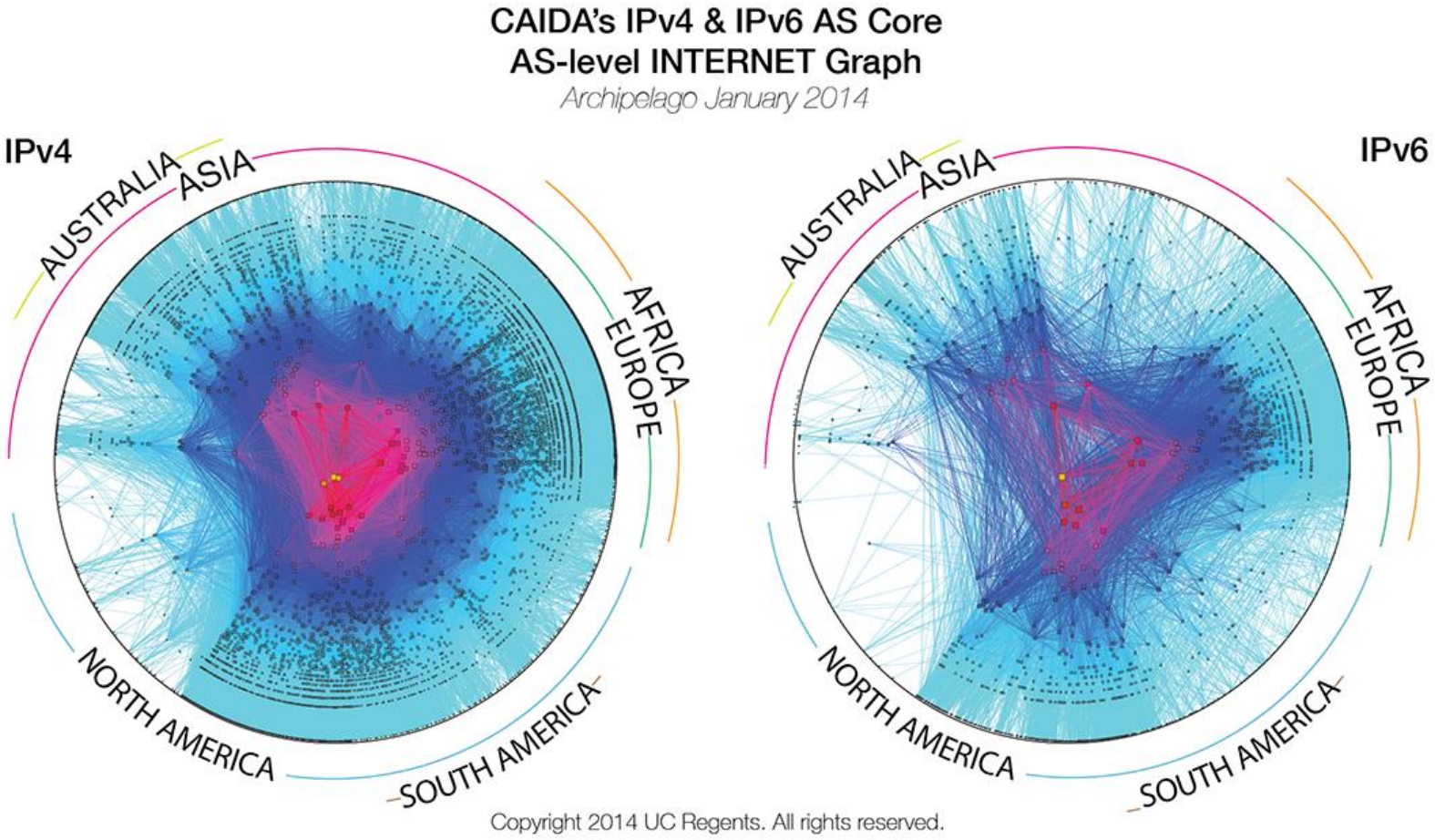
Die Icann ist die Hüterin über die Adressen im Internet. Bisher stand die Adressverwaltung unter der Aufsicht des US-Handelsministeriums. Jetzt stimmte das Gremium für neue Kontrollmechanismen.

http://www.spiegel.de/netzwelt/netzpolitik/icann-sagt-sich-von-us-abhaengigkeit-los-a-1081731.html

Historischer Hintergrund

- **Ursprünglich**: hauptsächlich Universitäten und Forschungseinrichtungen brauchen Internetzugang
- **Wandel**: Internet als Massenkommunikationsnetz
  ➔ 2^{32} IP-Adressen werden nicht (lange) reichen
- **Kurzfristig**: Entwicklungen und Maßnahmen, um mit den bestehenden IPv4-Adressen auszukommen (NAT, CIDR, ...)
- **Langfristig**: Migration zu IPv6 (128-bit Adressen)

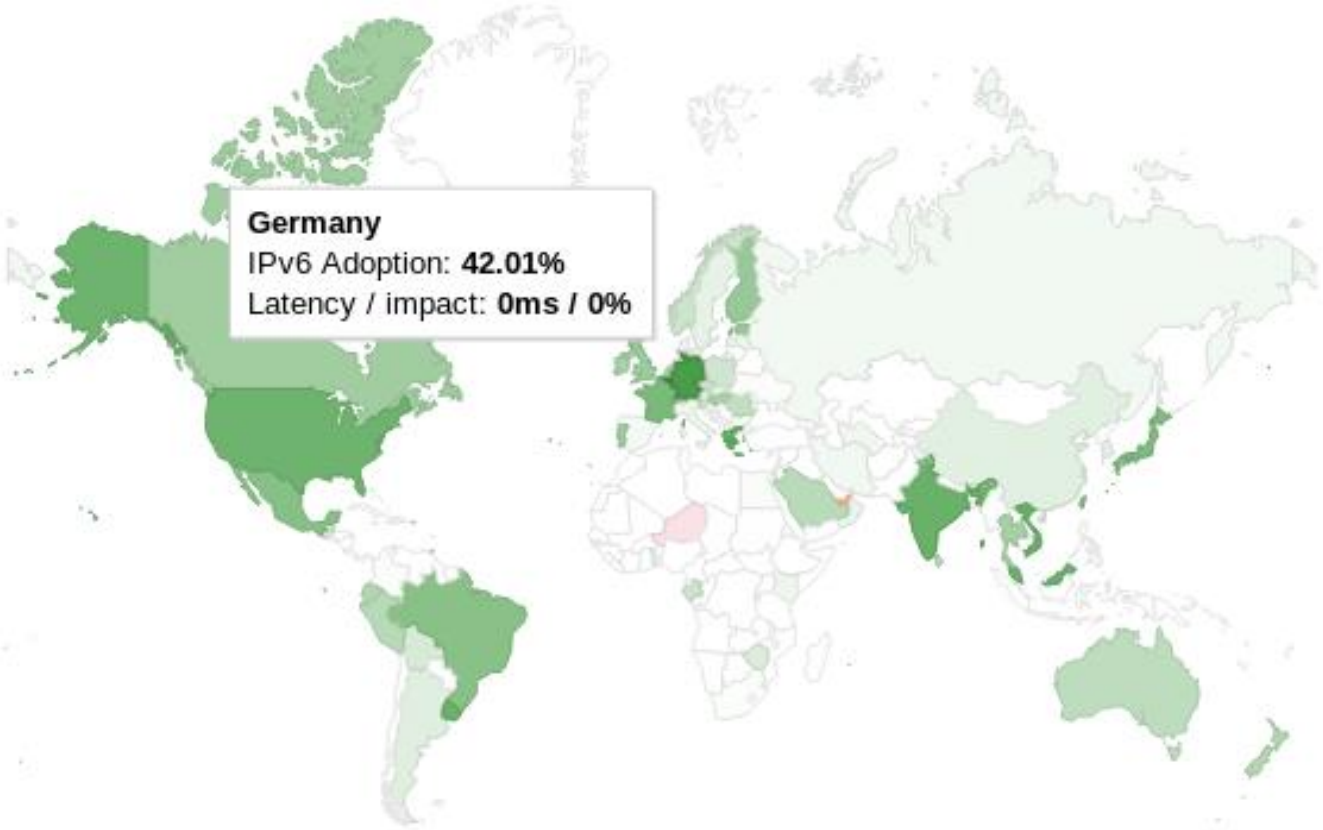
Karte des Internet (2014)



https://www.caida.org/research/topology/as_core_network/2015/

IPv6 Einführung pro Land

https://www.google.de/ipv6/statistics.html#tab=per-country-ipv6-adoption



Notation von IPv4-Adressen

- Durch Punkte getrennte, byteweise Dezimalschreibweise: p.q.r.s
- wobei p,q,r,s Dezimalzahlen zwischen 0 und 255 sind.
- Beispiel: „208.77.255.0“

Kapitel 3.3
DNS – Domain Name System

Namen im Internet

Einordnung

- IPv4-Adressen für (menschliche) Endnutzer „*ungünstig*“
- IP-Adressen sind (wegen hierarchischem Routing) an die Netztopologie gebunden. Wenn ein Host (oder Inhalt) also von einer Stelle im Internet an eine andere verlegt wird, ändert sich (in der Regel) die IP-Adresse.
- Gesucht: ein System um Hosts Namen zu geben, und diese Namen auf IP-Adressen abzubilden.

  ➔DNS (Domain Name System)
- Andere Ansätze: Aliasing, X.500, LDAP

Rahmenbedingungen

- Im LAN einer einzigen Organisation wäre es möglich eine zentrale Liste von Hostnamen mit zugehörigen IP-Adressen zu führen
- Im Internet widerspricht ein zentralverwalteter Ansatz der Grundidee (und ist technisch problematisch)

- 1.012.695.272 Hosts im DNS (Stand Januar 2019)

https://ftp.isc.org/www/survey/reports/current/

Herausforderungen bei der Namensvergabe

- Unabhängigkeit der Akteure
  - Kein globaler einheitlicher Zustand
  - Umziehen eines einzelnen Hosts muss allen mitgeteilt werden
- Skalierbarkeit
  - Abbildung von Namen auf Adressen muss für alle 2^{32} IPv4-Adressen bzw. 2^{128} IPv6-Adressen skalieren
  - Portierbarkeit auf nachfolgende größere Adressräume

Lösungsansätze

- Viele unabhängige Akteure:
  DNS-Namensraum als Baumstruktur ➔ hierarchischer Namensraum
  - Einzelne Äste/Bereiche des Baums werden einzelnen Akteuren zugesprochen.
- Skalierbarkeit (große und wachsende absolute Zahl an Teilnehmern):
  Implementierung als verteilte Datenbank.
  - Bei Wachstum des Systems (bzw. Last) können einfach weitere DNS Server hinzugefügt werden.

Überblick DNS

DNS ...

- ist ein Dienst der Anwendungsschicht
- ist als verteilte Datenbank mit einer Hierarchie von Nameservern implementiert.
- bietet einen als Baum strukturierten (hierarchischen) Namensraum für Hosts im Internet.
- ist kritischer Bestandteil des Internets!

DNS als Dienst

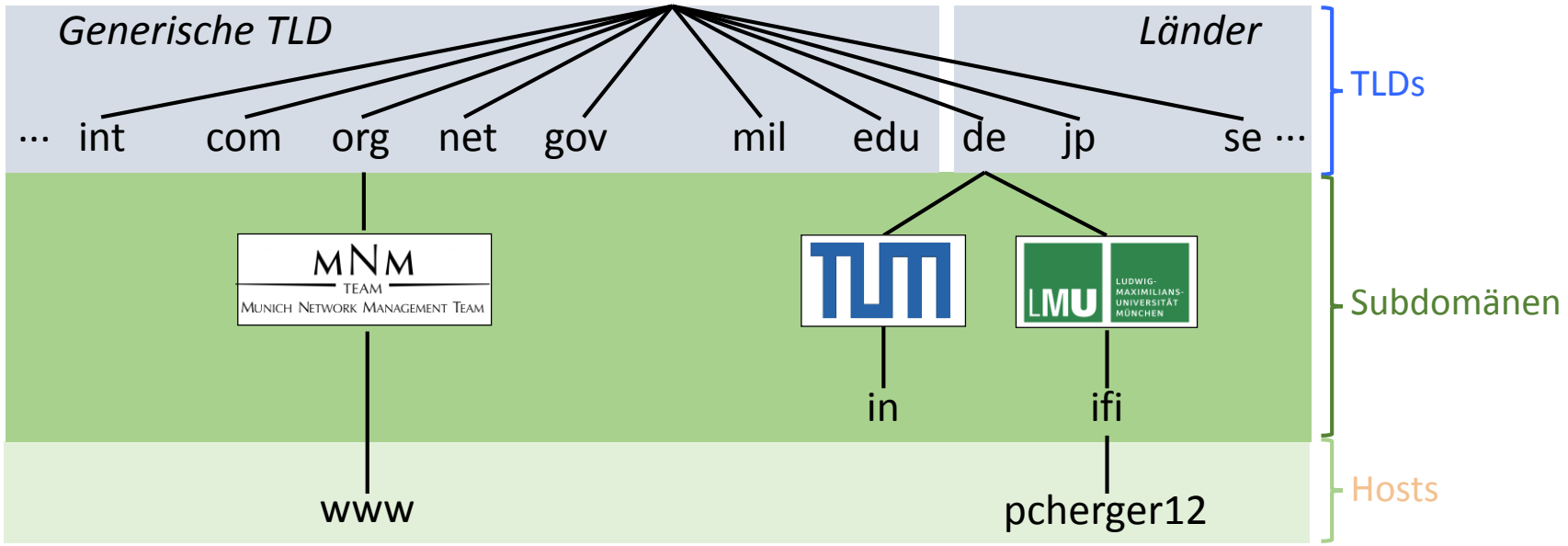
- Dienste und Dienstmerkmale:
  - Namensauflösung (Abbildung Host-Namen auf IP-Adressen
    - *Fully Qualified Domain-Name (FQDN)*
    - *Resource Records = <name> [<ttl>] [<class>] <type> <rdata>*
  - Aliasing (insbesondere für Hosts, Mail Server)
  - Redundanz
  - Lastverteilung zwischen replizierten Servern
- Grundfunktionen eines DNS-Servers:
  - Beantworten von Client-Anfragen
  - Austausch mit anderen DNS-Servern

Der DNS Namensraum (1)

- Unterscheidung: Generische und Länder-Domänen
- Top Level Domains (TLD) von der ICANN (Internet Corporation for Assigned Names and Numbers) verwaltet
- Für jeden TLD gibt es einen Registrar, der (gegen Gebühr) second level Domains in diesem TLD an verschiedenste Organisationen (z.B.: LMU) sowie Privatleute vergibt (➔ First Come, First Served)

Der DNS Namensraum (2)

- Second Level Domains (und tiefer) können von den Organisationen, denen sie zugesprochen wurden, beliebig in weitere Subdomänen eingeteilt werden. (z.B.: „ifi“ als Subdomäne von „lmu“)
- Blätter des Baums enthalten Hostnamen (In der Praxis kann „Hostname“ mit vielen IP-Adressen /tatsächlichen Hostmaschinen assoziiert werden)
- Der *Fully Qualified Domain Name* (FQDN) ist der volle Pfad von einem Blatt des Baums bis zur Wurzel (durch Punkte getrennt) z.B.: „pcheger12.nm.ifi.lmu.de.“ (Die Wurzel nach dem letztem Punkt ist namenslos)



- Hierarchisch, Baumstruktur
- Top Level Domains (TLD): festgelegt von ICANN
  - generisch, nach Zweck (gTLD)
  - TLD für Länder (ccTLD, ca 240 Stück)
  - seit 2013: „new gTLDs“
- Subdomains: Unterteilung in benannte Teildomänen
  - Second-level domains (z.b. `lmu.de`) von Registraren zugeteilt
  - Unterteilung in Subdomains kann wiederholt werden
- Host-Name: Blätter des Baumes
- Fully Qualified Domain Name (FQDN)
  - vollständiger Name bestehend aus Hostname und Domänennamen
  - in Richtung Baumwurzel zu lesen; endet mit Punkt

Cybersquatting

- Engl. Squatter = Hausbesetzer
  - Domänenbesetzung, Domainsquatting, Namejacking, Brandjacking
- Registrierung von Domain-Namen, die für andere Personen oder Institutionen von Interesse sind
  - Markennamen, Musiker, Sportler, ...
- Verkauf der Namen
- Rechtliche Situation

Ressourcendatensätze (Engl.: Resource Records)

- Die durch DNS implementierte Datenbank enthält als Daten sogenannte Resource Records (RR)
- Jede Domäne kann mit beliebiger Anzahl von RRs assoziiert sein
- Resource Records sind (zur Effizienz) binär codierter Fünftupel mit folgendem Schema: (<Domain Name>,<TTL>,<Klasse>,<Typ>,<Wert>)

Resource Records

- *Resource Records* sind ein (zur Effizienz) binär codierter Fünftupel mit folgendem Schema: (<Domain Name>,<TTL>,<Klasse>,<Typ>,<Wert>)
  - Der **Domain Name** ist normalerweise der primäre Suchschlüssel für die Anforderung von RRs (z.B.: „lmu.de“).
  - **TTL** (engl.: time to live) ist die Zeit in Sekunden für die, die Instanz des RR als gesichert (wahrscheinlich korrekt) gilt.
  - Das **Klasse** Feld enthält in der Praxis fast immer den wert „IN“ für „Internet“.
  - Eine Auswahl von RR **Typen** gibt es auf der nächsten Folie.
  - Der **Wert** sind die eigentlichen Daten des RRs.

Wichtige RR Typen

Typ	**Bedeutung**	**Zugehöriger Wert**
SOA	Start of Authority	Infos zur Zonen Verwaltung
A	IPv4 Address Record	32-bit Integer, IPv4-Adresse
AAAA	IPv6 Address Record	128-bit Integer, IPv6-Adresse
MX	Mail Exchange Record	Zuordnung der zuständigen Mailserver
NS	Nameserver Record	Hostname eines autoritativen Nameservers
CNAME	Canonical Name Record	Kanonischer Name eines Hosts
PTR	Pointer	Für Reverse Mapping: IP-Adressen➔Namen
TXT	Text Record	Ursprünglich frei definiert, heute SPF (Sender Policy Framework), DomainKeys, DMARC, DNS-SD, Google-Site Verificiation

Zonen im DNS Namensraum

- DNS Namensraum wird in **nicht überlappende administrative Zonen** aufgeteilt
- Administratoren einer Zone stellen für Anfragen zu beliebigen FQDNs in ihrer Zone „**autoritative Nameserver**“ bereit

- Anmerkung: Die Administratoren einer Zone sind nicht unbedingt identisch mit den Inhabern der zugehörigen Domänen

Autoritative Nameserver

- Antworten von „*autoritativen Nameserver*“ gelten als richtig
- Allgemein: zur **Lastverteilung/Ausfallsicherheit mehrere autoritative Nameserver** pro Zone bereitgestellt
  - Ein primärer Nameserver (wird aktiv gewartet)
  - Alle anderen sind sekundäre Nameserver (holen regelmäßig aktuelle Informationen vom primären Nameserver per „Zonentransfer“)
  - Primäre und sekundäre Nameserver einer Zone gelten gleichermaßen als autoritativ.

Weitere Nameserver-Typen

- Root Nameserver
  - 13 Root-Nameserver weltweit
  - gut geschützte stark replizierte Hochleistungssysteme (werden per *anycast routing* angesteuert)
  - kennen vor allem die autoritativen Nameserver der top-level Domänen (sowie die populärer second-level Domänen)
- Lokale Nameserver
  - Hosts im Internet müssen mindestens einen Nameserver mit IP-Adresse kennen, um überhaupt eine Anlaufstelle für die Namensauflösung (und damit das Ermitteln weiterer IP-Adressen) zu haben ➔ Lokaler Nameserver des Hosts
  - Werden vor allem von ISPs bereitgestellt.

Anfragen an DNS - Ablauf

1. **Anfrage** an Resolver (lokales Programm des Hosts)
2. Nachschlagen im Cache – Ergebnis? (Ja / Nein)

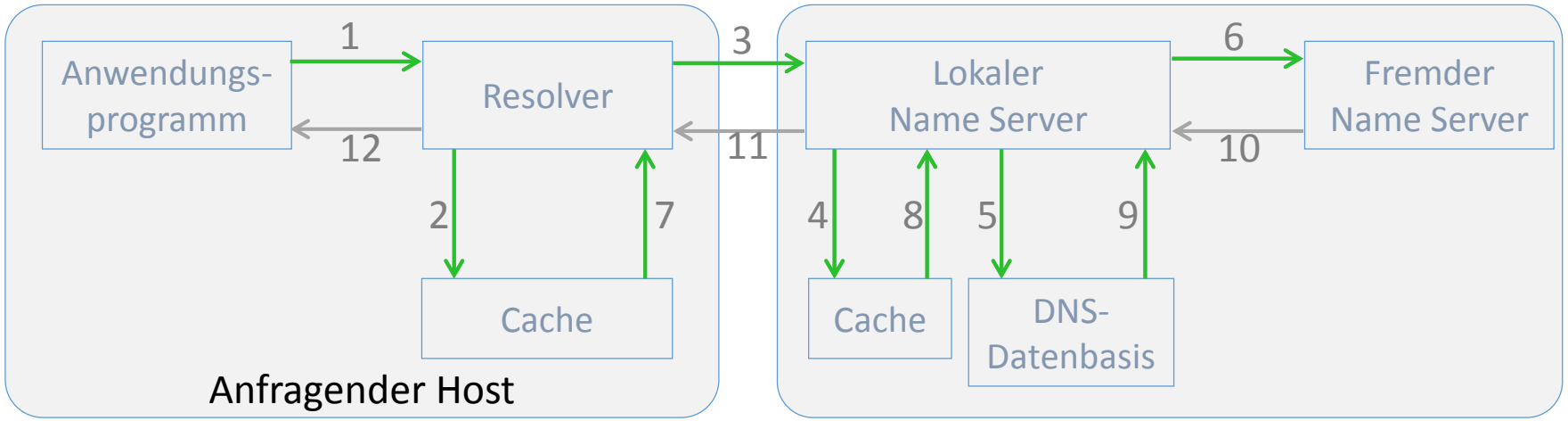
Nein **3. Anfrage** an lokalen Nameserver

4. Nachschlagen im Cache – Ergebnis? (Ja / Nein)

Nein 5. Nachschlagen in RRs des lokalen NS – Ergebnis? (Ja / Nein)

Nein **6. Anfrage** an fremde Nameserver – Ergebnis?

Achtung: RRs in einem Cache sind niemals autoritativ und werden nach Ablauf der *time-to-live* (TTL) verworfen!



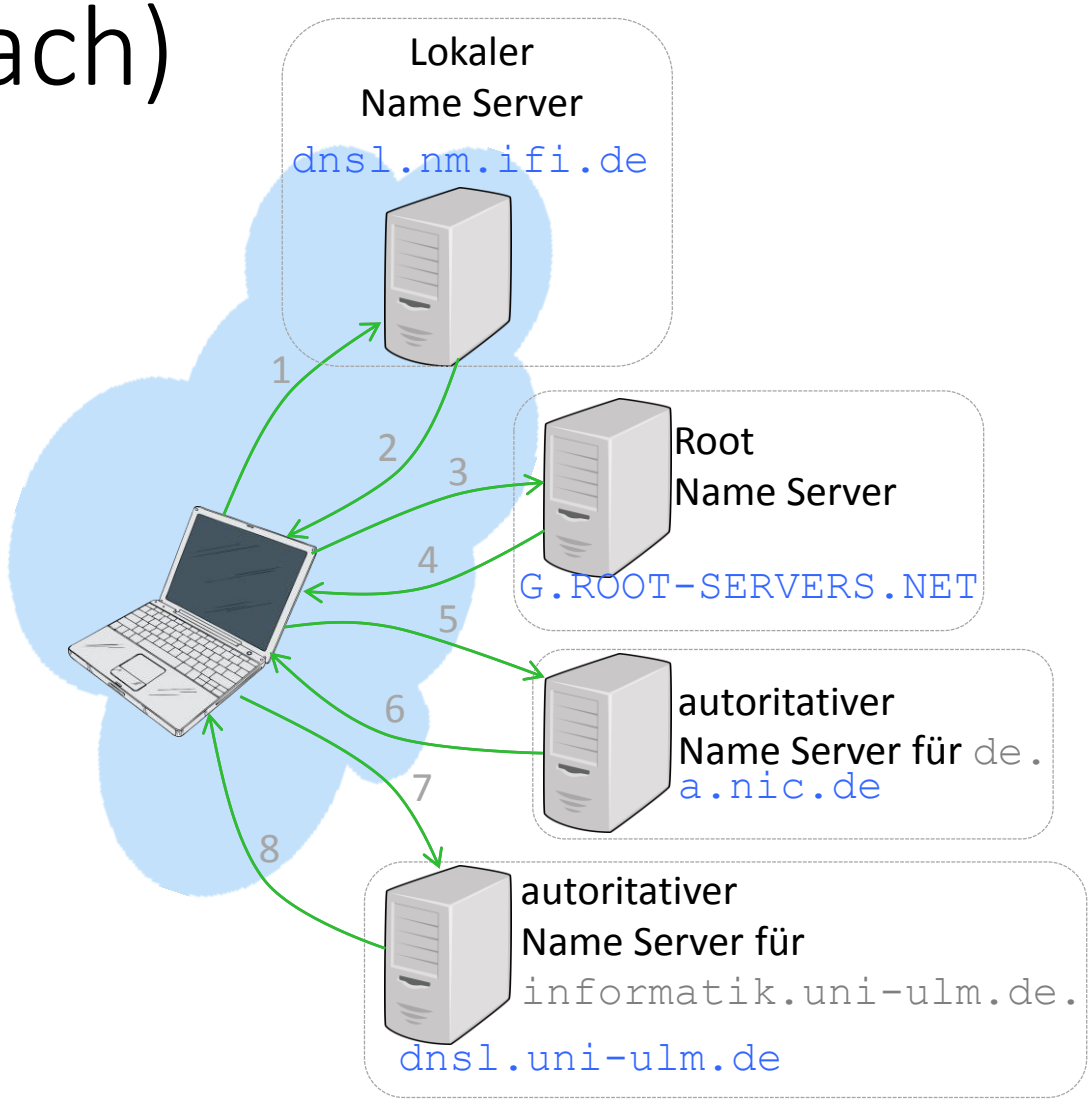
- Schritte
  1. Anfrage an den Resolver (lokal auf Host)
  2. Nachschlagen im Cache.(Bei Erfolg: 7, 12)
  3. Anfragen bei lokalem DNS-Server
  4. Nachschlagen in Datenbasis.(Bei Erfolg: 9, 11, 12)
  5. Nachschalgen im Cache.
  6. Anfrage an fremden DNS-Server. Antwort: 10, 11, 12
- Inhalt der Antwort (Bei Erfolg: 8, 11, 12 + Update Resolver-Cache)
  - gesuchte IP-Adresse, oder
  - Referenz auf DNS-Server, der den Name auflösen kann, oder
  - „gibt es nicht“
- Antwort führt zur Auffrischung der Cache-Information

Abarbeitung der DNS Anfragen

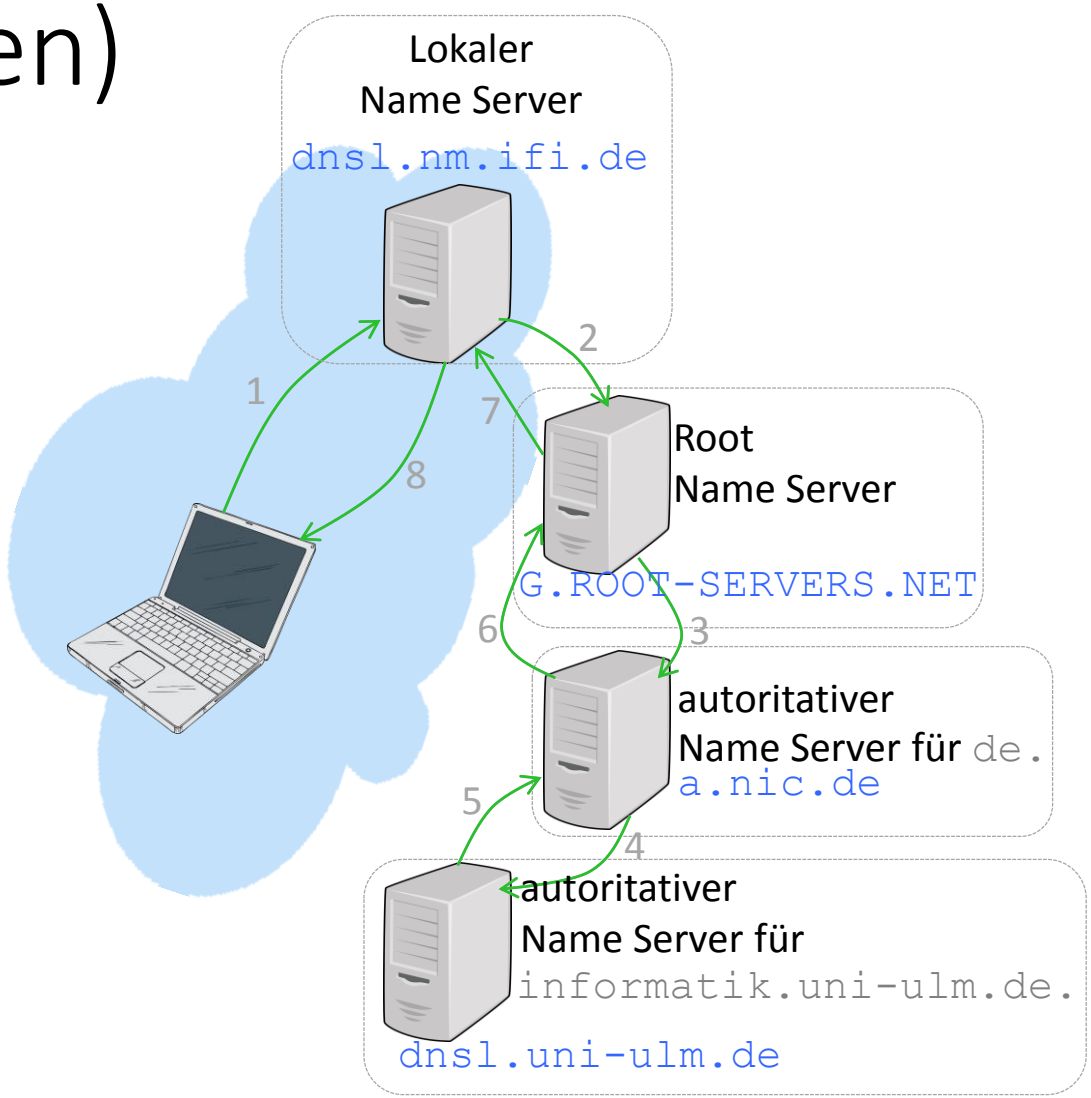
- Man unterscheidet
  - Iterative Anfragen
  - Rekursive Anfragen

- Iterative Anfragen geben jeweils Teilantwort der Anfrage (z.B.: Root Nameserver gibt nur den NS für „.de“ als Antwort zur Anfrage „dns1.nm.ifi.lmu.de“)
- Rekursive Anfragen geben immer vollständige Antwort, erhöhen aber die Last auf beteiligte Nameserver (außerdem Sicherheitsrisiken)

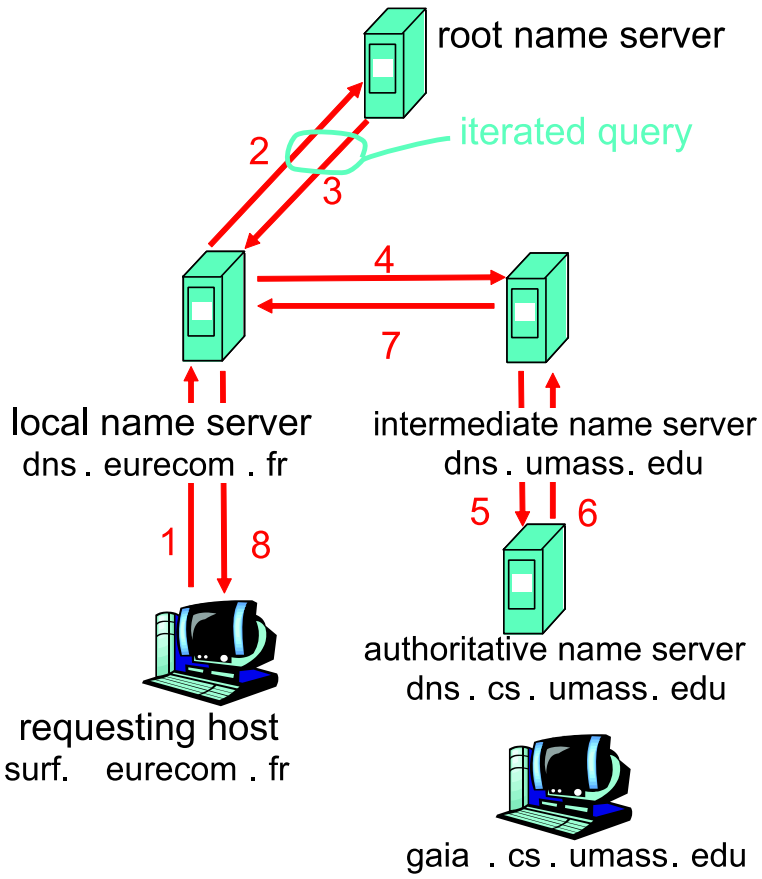
DNS: Iterative Anfrage (Der Reihe nach)



DNS: Rekursive Anfrage: (Durchreichen)



Regelfall: kombinierter Ansatz



- In der Praxis meist Kombination aus rekursiven/iterativen Anfragen. (Sofern die gewünschte Information nicht schon in einem Cache vorhanden ist)
- Root Nameserver beantworten Anfragen im allgemeinen nicht rekursiv.
- Hosts stellen allgemein rekursive Anfragen an ihren lokalen Nameserver. Dieser arbeitet sie iterativ ab.

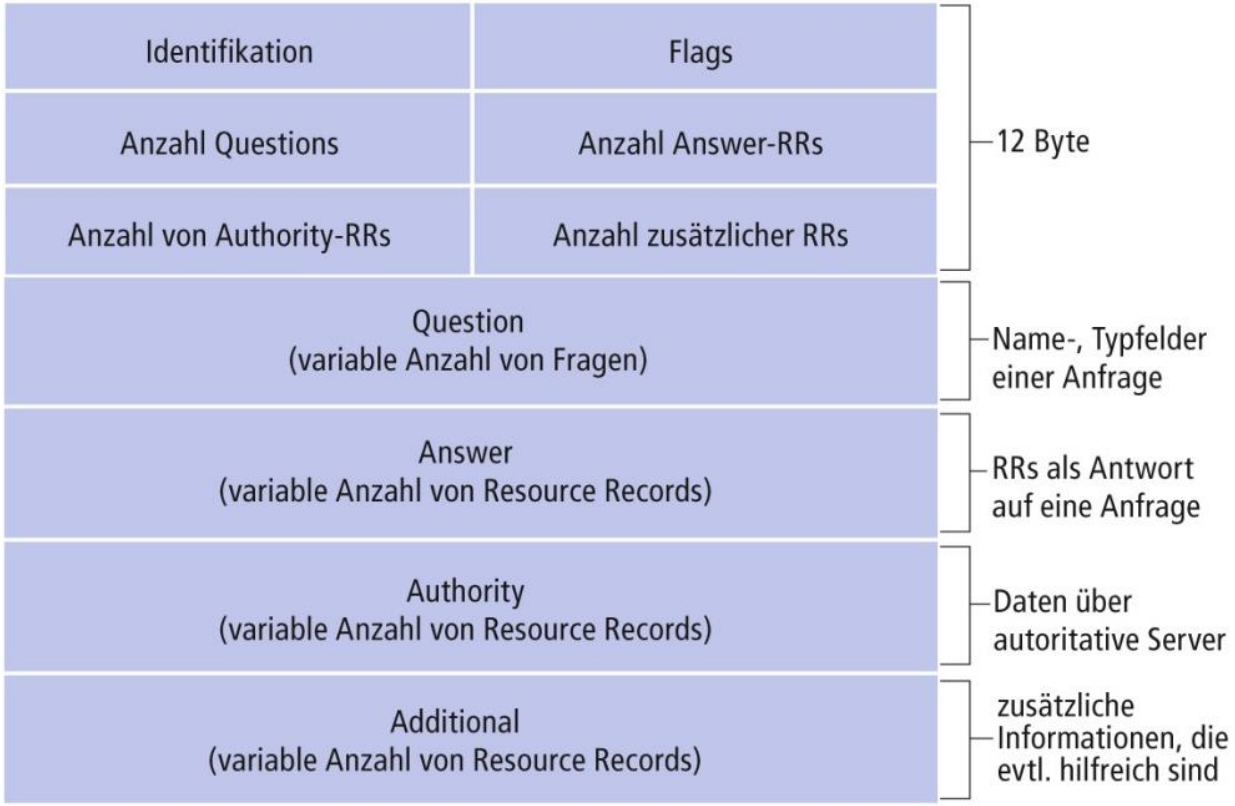
Beispielanfrage: host und dig

```
danciu@pcheger09:~> host www.in.tum.de
www.in.tum.de is an alias for www.informatik.tu-muenchen.de.
www.informatik.tu-muenchen.de is an alias for infoport.informatik.tu-muenchen.de.
infoport.informatik.tu-muenchen.de has address 131.159.74.65
infoport.informatik.tu-muenchen.de mail is handled by 25 mailin.informatik.tu-muenchen.de.
```

```
; <<>> DiG 9.3.4 <<>> www.in.tum.de
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 7702
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 5, ADDITIONAL: 7
;; QUESTION SECTION:
;www.in.tum.de.                 IN      A
;; ANSWER SECTION:
www.in.tum.de.          86400   IN      CNAME   www.informatik.tu-muenchen.de.
www.informatik.tu-muenchen.de. 86400 IN CNAME   infoport.informatik.tu-muenchen.de.
infoport.informatik.tu-muenchen.de.     86400   IN      A       131.159.74.65
[...]
```

Name — TTL — Class — Type — Value

DNS-Nachrichtenformat



- Identification: Anfrage ID
- Flags: Query/Reply, Authoritative Bit, Recursion Desired, Recursion Available

Wahl des Transportprotokolls

- Anfragen: UDP-basiert (Performanz)
  - Verzicht auf Verbindungsaufbau
  - Anfragen zustandslos
  - Wiederholung möglich, falls Frage oder Antwort verloren gehen
- Zonentransfer: TCP-basiert (Zuverlässigkeit)
  - Größeres Datenvolumen als einzelne Anfrage
  - Findet im Vergleich zu Anfragen selten statt
- Details zu TCP/UDP in nachfolgenden Veranstaltungen

Fragen zu DNS

- Wie unterscheiden sich die Rollen eines lokalen und eines autoritativen Nameservers? Kann ein einziger Server beide gleichzeitig erfüllen?
- Was sind die Unterschiede zwischen einer DNS Domäne und einer DNS Zone?
- Was unterscheidet eine iterative von einer rekursiven DNS-Anfrage? Können beide kombiniert werden?

Klausurtermin

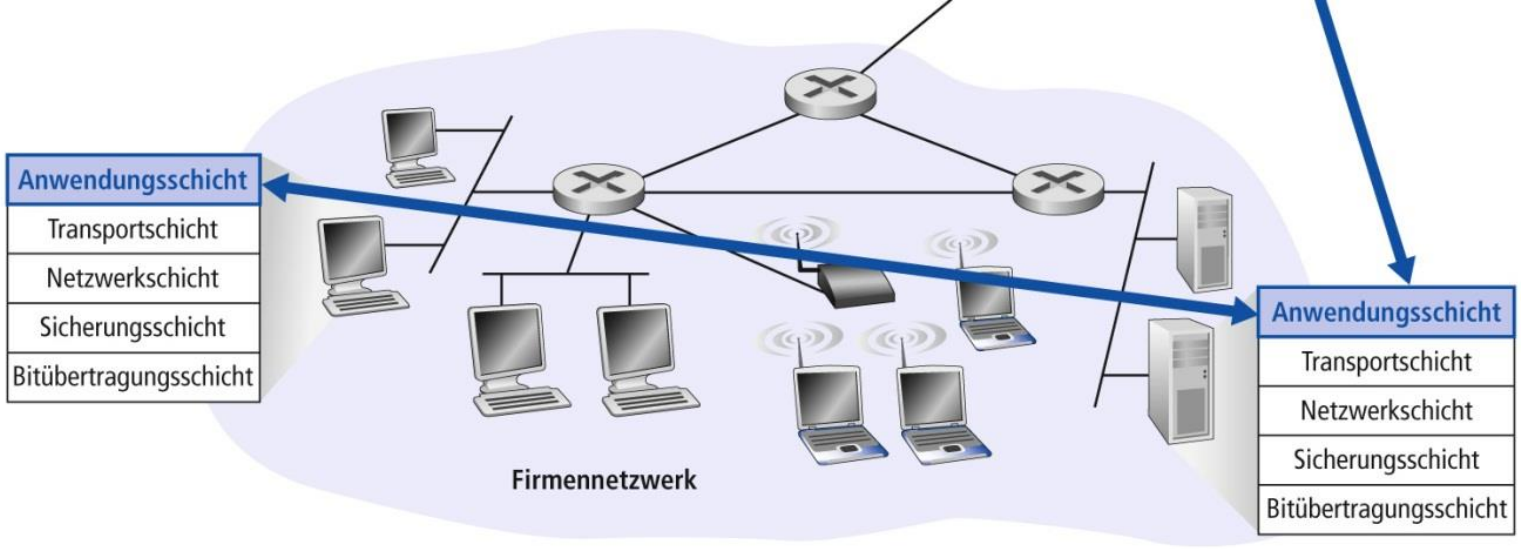
Achtung: Änderungen beim Klausurtermin!

Semesterklausur wegen Kollisionen verschoben auf:

Donnerstag, **01.08.2019** von 18:00 bis 20:00 Uhr

Nach wie vor im Hauptgebäude der LMU. Die genaue Raumeinteilung wird vor der Klausur bekannt gegeben.

Sie müssen sich im Uniworx **erneut zur Klausr anmelden**!



Kapitel 3.4 Architekturen verteilter Anwendungen

Wie kann eine Anwendung im Netz aufgebaut werden?

Anwendungsarchitektur

Achtung: Architektur der Anwendung != Netzarchitektur

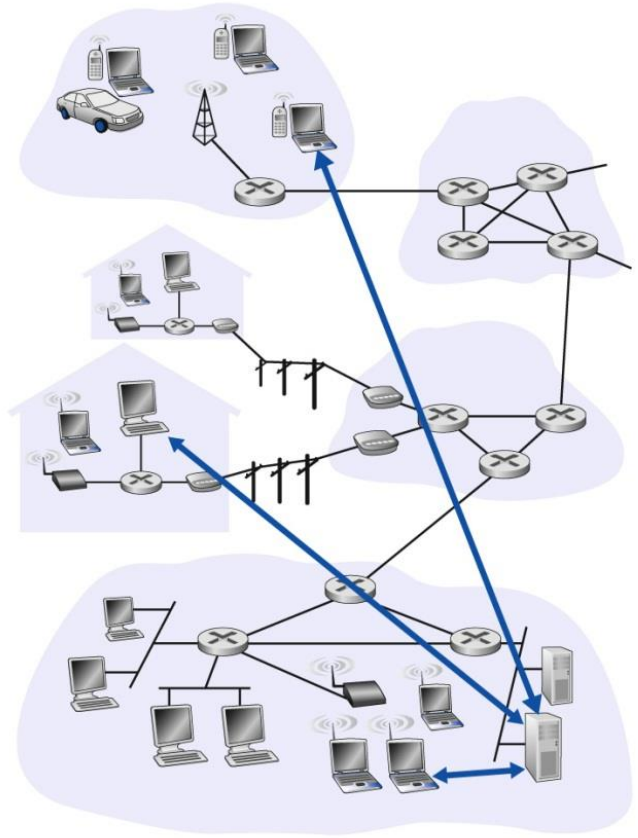
Die Anwendungsarchitektur definiert die Interaktionsstruktur aus Anwendungssicht.

Hauptsächlich gibt es zwei Paradigmen:

- Client-Server (zentral)
- Peer-to-Peer (dezentral)

Client-Server

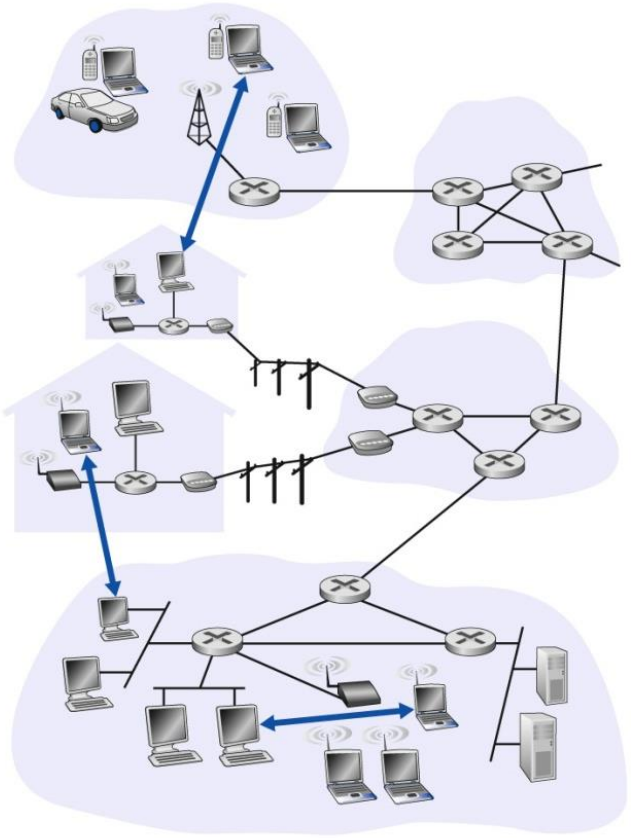
- Server: ständig erreichbar
- Client: sendet Anfragen an Server
- Beispiele: Web, E-Mail, Telnet, SSH



a Client-Server-Architektur

Peer-to-Peer-Architektur (P2P)

- Kaum oder keine Abhängigkeit von Servern → dezentral organisiert
- Direkte Verbindung von *Peers*
- Beispiele: BitTorrent, IPFS (verteiltes Web), Bitcoin, Tor



b Peer-to-Peer-Architektur

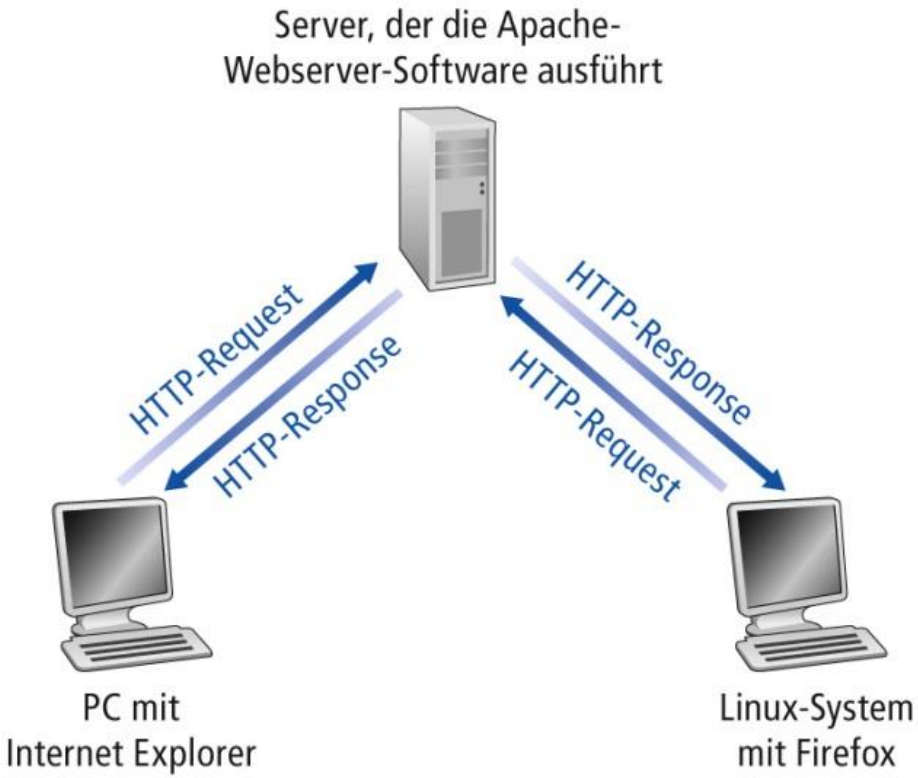
Kapitel 3.5

Protokolle der Anwendungsschicht

Web: Hypertext Transfer Protocol

1. Client sendet einen HTTP-Request an den Webserver
2. Server antwortet mit einer HTTP-Response

Sowohl Request als auch Response enthalten Steuer- und Nutzdaten



Grundlagen (1/2)

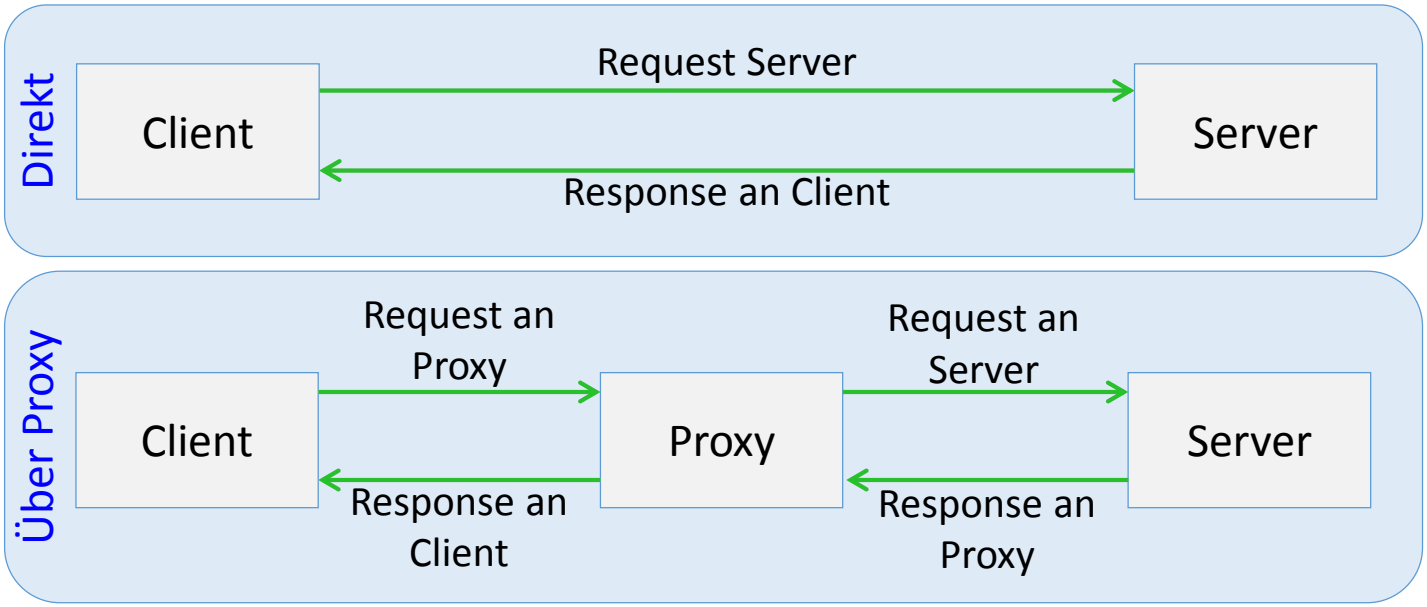
- Das World Wide Web (kurz Web) ist ein System durch *Hyperlinks* verknüpfter Web-Ressourcen (Webseiten/Hypertext-Dokumente), welches über das Internet zugänglich ist (mit Hilfe von Protokollen wie HTTP).
- HTML (engl. Hypertext Markup Language) zur Formatierung von Dokumenten
- URL (engl. Uniform Ressource Locator) zur Referenzierung von Objekten (Adressierung)
- HTTP (engl. Hypertext Transfer Protocol) zur Übertragung über das Internet

Grundlagen (2/2)

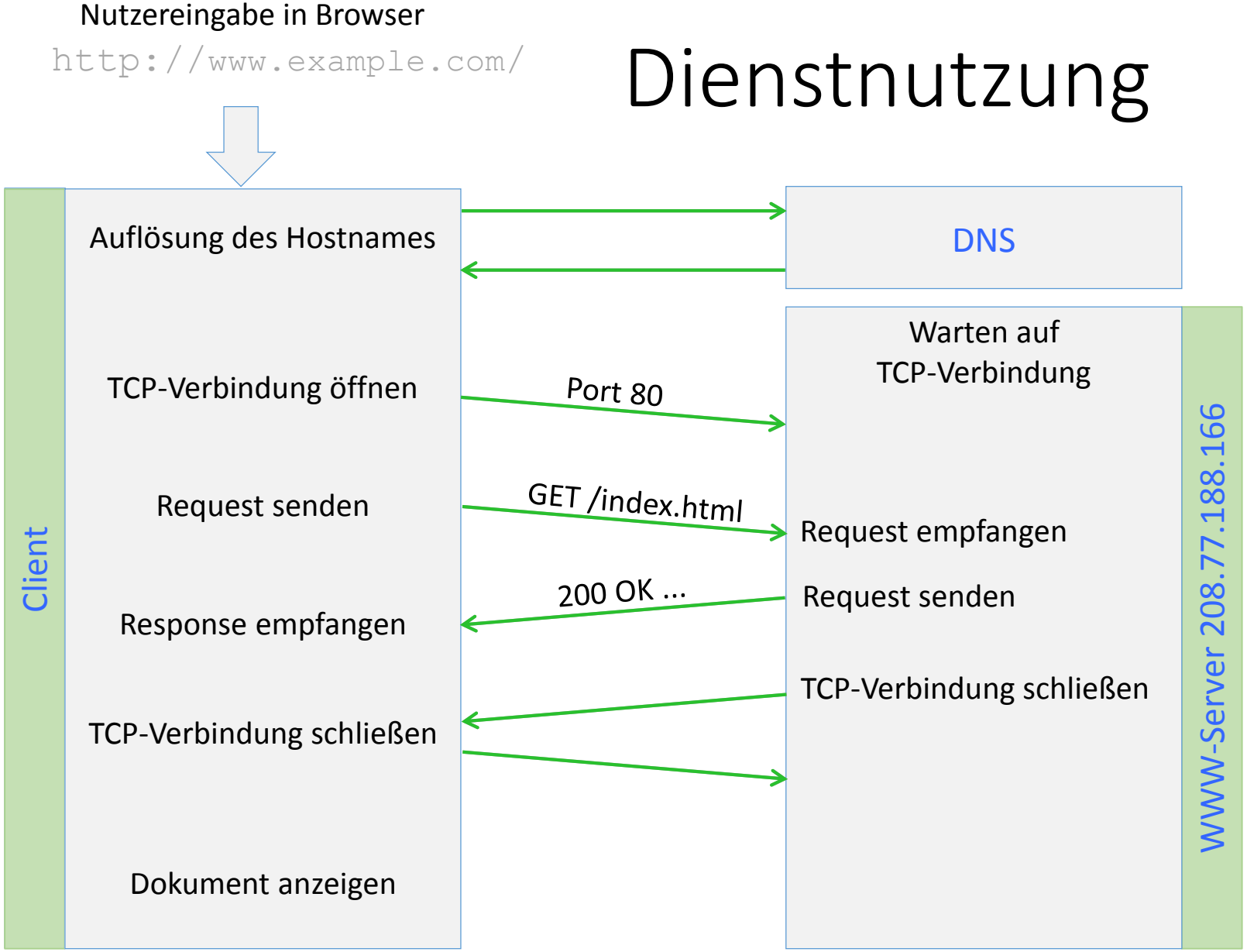
- Hypertext Markup Language (HTML): Beschreibungssprache für Web-Dokumente
  - standardisiert als W3C Recommendation, z.B. HTML 5
  - Dokumente bestehen aus Header (Titel, Formatierung, Metadaten) und Body (Inhalt)
  - Links durch Anker (anchor) angegeben, die eine URL enthalten.
  - Hilfen: Cascading Style Sheets (CSS), Scalable Vector Graphics (SVG)
- Referenzierung von Objekten (Adressierung)
  - **URI** (U. R. Identifier) RFC 1630: Objektbegriff für URL und URN
  - **URN** (U. R. Name): global eindeutiger langlebiger logischer Name für Objekt (ohne Lagerort)
  - **URL** (Uniform Resource Locator) RFC 1738: Lagerort eines Web-Dokuments durch Serverangabe und Pfadbezeichnung
  - URL Syntax:
    <protokoll>**://**[<user>[**:**<passwd>]**@**]<host>[**:**<port>]**/**[<path>]
  - z.B. http://www.example.com/documents/index.html
  - Optionale (und eher seltene) Felder: user, passwd, port

Hypertext Transfer Protocol (HTTP)

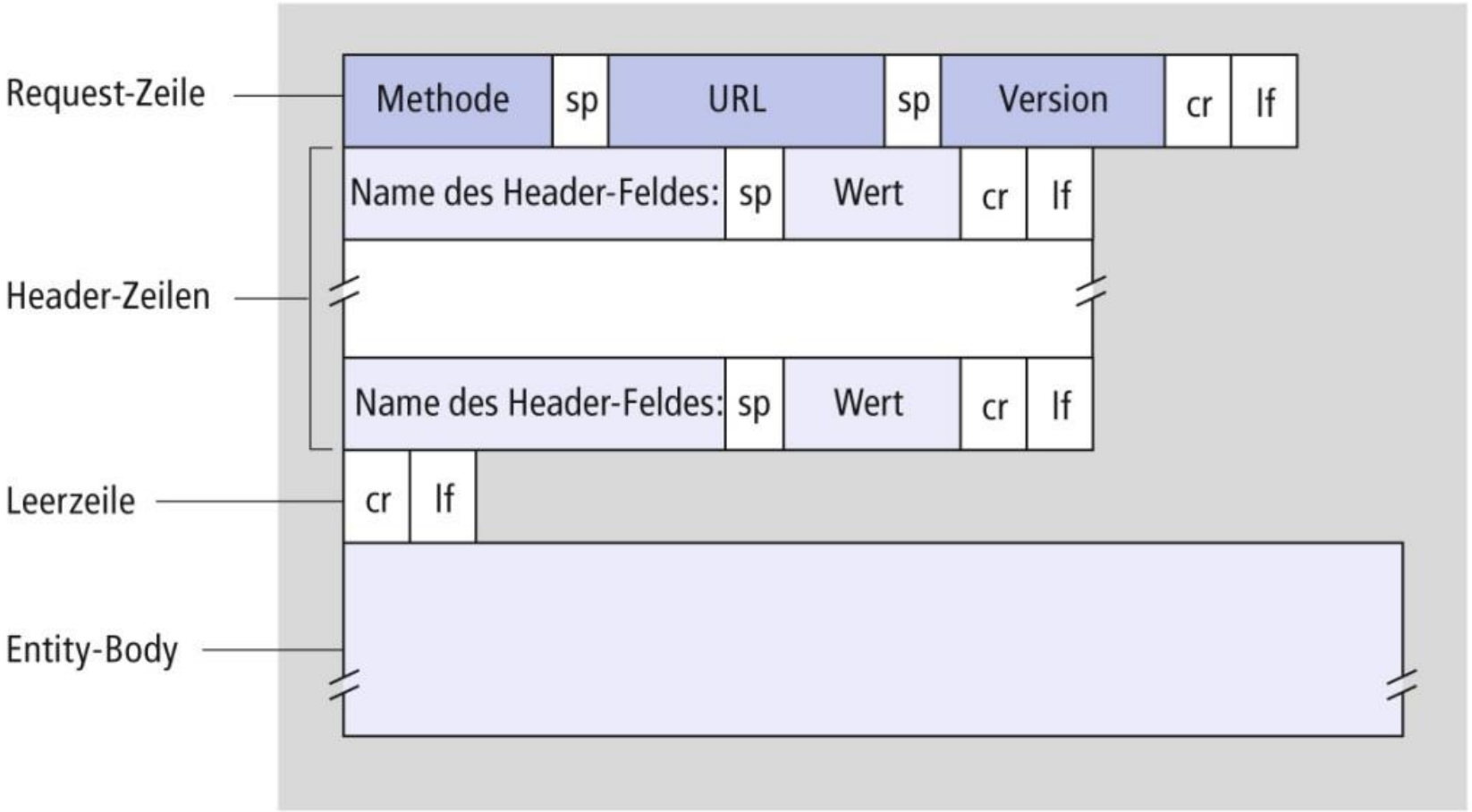
- Protokoll zum Transport von Anfragen/Objekten zwischen Browser, Server und Proxy (Zwischensystem)
- TCP-basiert, Port 80 (Proxy: typischerweise Port 8080)
- HTTP ist zustandslos, pull-orientiert, unterstützt bidirektionale Übertragung und Caches im Client bzw. Proxy
- HTTP/1.1 spezifiziert in RFC 2068, 2616



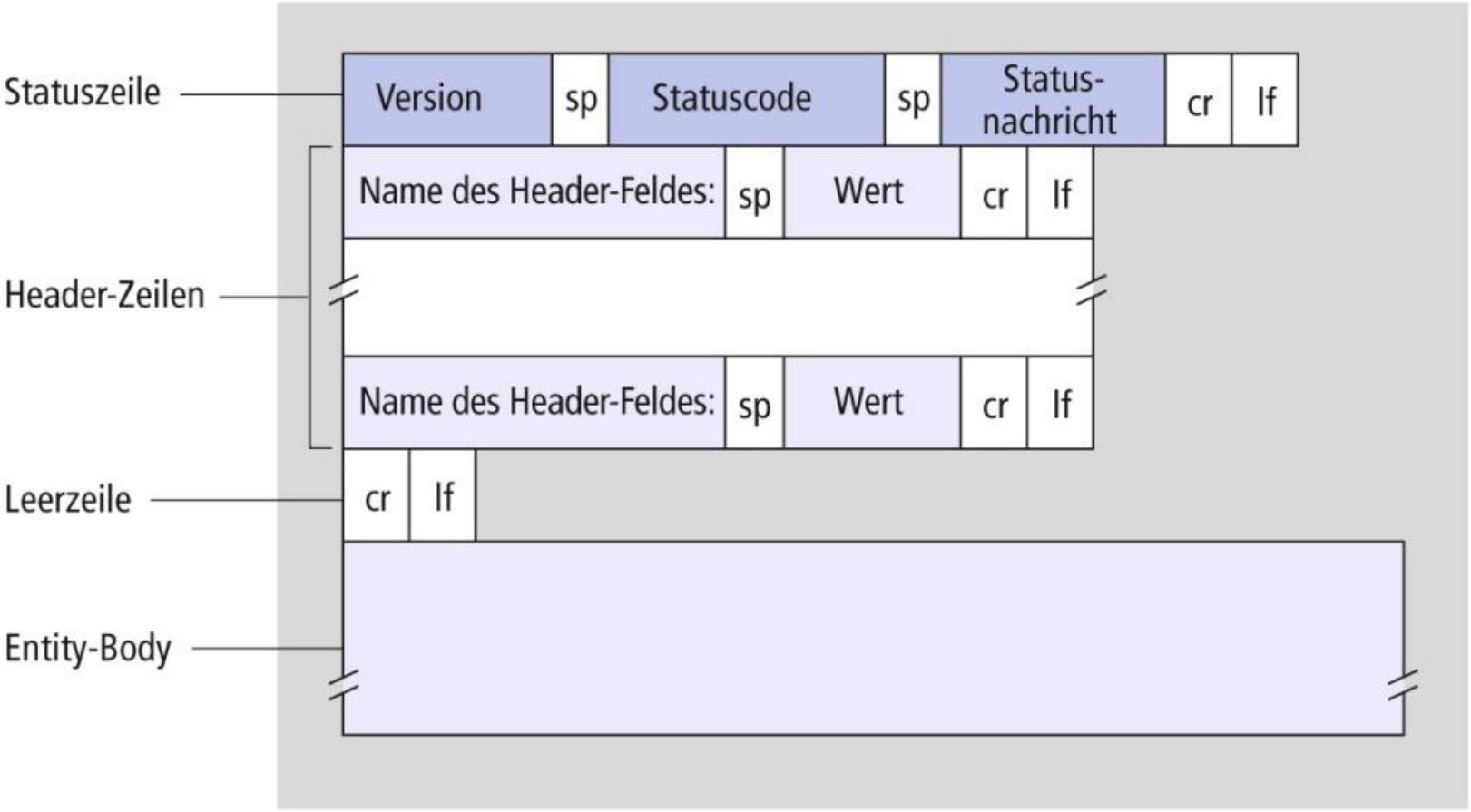
Dienstnutzung



HTTP-Request



HTTP-Response



Request-Methoden und Statuscodes in HTTP/1.1

Methoden/Dienstprimitive

- GET: Abruf einer Datei vom Server.
- HEAD: Abruf von Metadaten über eine Datei.
- POST: Übertragen von Daten an den Server.
- PUT: Ablegen einer Datei auf Server
- DELETE: Löschen einer Datei auf dem Server.
- OPTIONS: Abfrage von Informationen über Kommunikationsoptionen.
- TRACE: für Testzwecke.

Statuscodes

- 1xx: Informational
  100 Continue
  101 Switching Protocols
- 2xx: Successful
  200 OK
  206 Partial Content
- 3xx: Redirection
  301 Moved Permanently
  302 Found
  307 Temporary Redirect
- 4xx: Client Error
  400 Bad Request
  401 Unauthorized
  403 Forbidden
  404 Not Found
- 5xx: Server Error
  500 Internal Server Error

HTTP-Request-Response Beispiel

```
> GET /index.html HTTP/1.1
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Thu, 10 May 2019 12:03:26
GMT
< Server: Apache/2.2.16 (Debian)
< Transfer-Encoding: chunked
< Content-Type: text/html;
charset=utf-8

<!DOCTYPE html>
<html>
…
```

Anfrage nach index.html
Clientsoftware ist das Programm „curl“
Als Inhaltstyp der Antwort wird alles akzeptiert (z.B. application/pdf)

Status 200 → Anfrage erfolgreich

Inhaltstyp der Antwort: HTML

HTTP als zustandsloses Protokoll

- Vorteile: einfach und fehlerunempfindlich
- Nachteil: viele Dienste benötigen Zustandshaltung
  - Dienstsitzung besteht aus **mehreren Request/Response-Paaren** (Schritten) — z.B. Buchung einer Fahrkarte
  - Dienststatus muss **über alle Schritte** erhalten werden
- Abhilfe (zum Standard erhobene Notlösungen)
  - Statusvariablen in URL werden per GET-Methode übertragen
  - Cookies: clientseitige Speicherung einer Zeichenkette
  - SessionID: Vergabe eindeutiger Kennung für eine Dienstsitzung (oft in URL); Status serverseitig gespeichert.

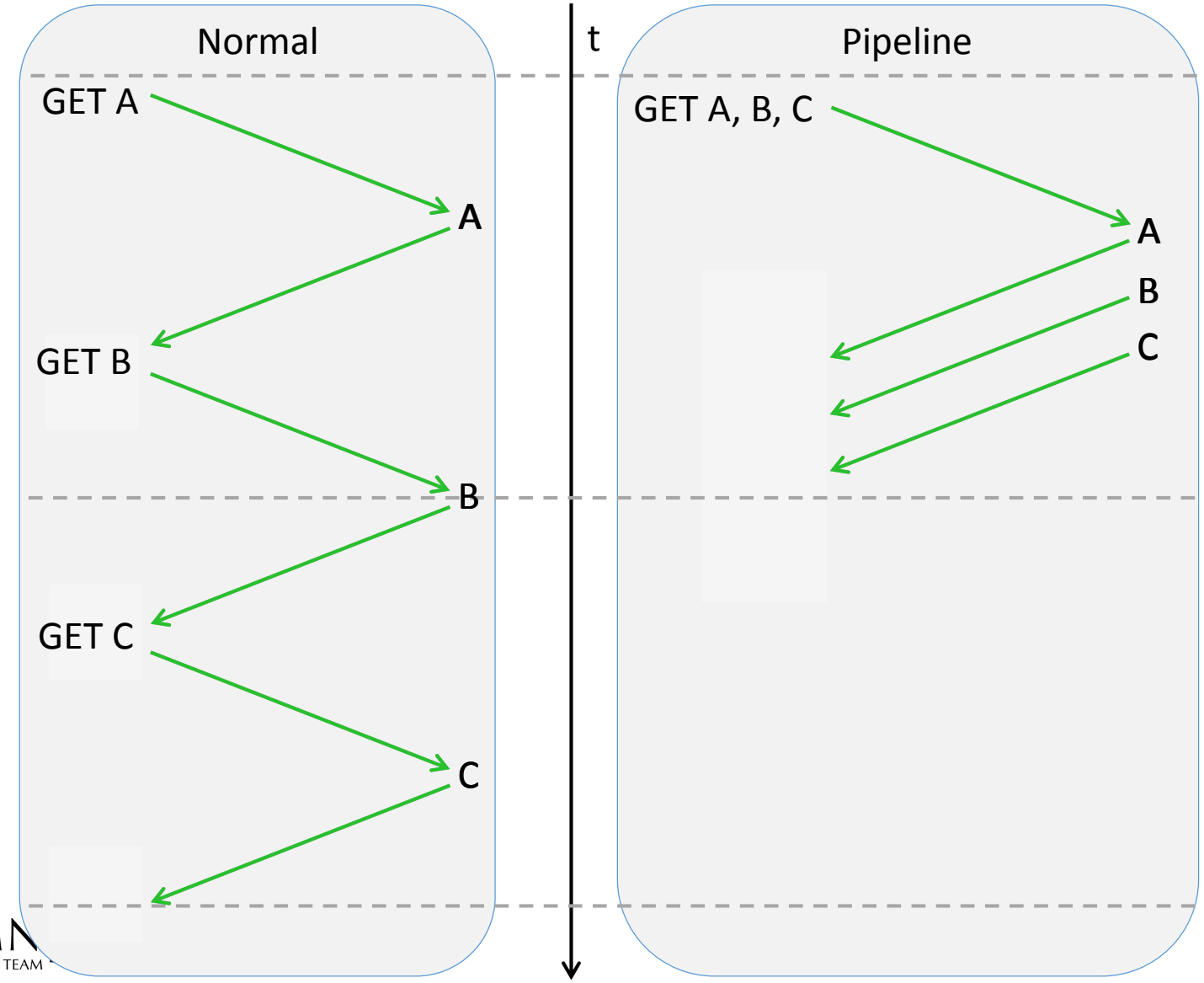
Sicherheit

- Authentifizierung
  - IP-Adresse (nicht praktikabel)
  - Kennung/Passwort (Basic Authentication) oder kryptographische Zertifikate
- HTTPS: Verschlüsselte Variante von HTTP (RFC 2660)
  - meist zusammen mit Serverauthentifizierung (Zertifikat)
  - Übertragung über Transport Layer Security (TLS), Secure Socket Layer (SSL)(Port 443)

Leistungssteigerung

- Ziele: schnelle Anzeige von Webseiten; geringe Netzlast
- Einfachster Fall
  - Eine TCP-Verbindung zum Server pro Request/Response-Paar
  - Nach Request wird auf Response gewartet.
  - Client ruft Objekt jedes Mal vollständig vom Server ab.
  - Verbindung wird nach Interaktion geschlossen.
- Maßnahmen zur Leistungssteigerung
  - Mehrere gleichzeitige Verbindungen: Parallelisierung der Anfragen
  - Persistente Verbindung: Verbindung wird für mehrere Anfragen wieder benutzt ➔ Einsparung des Verbindungsaufbaus („Connection: keepalive“)
  - Pipelining: Client schickt mehrere Requests nacheinander; Server schickt entsprechende Responses.
  - Caching: Client verwaltet lokales Cache jüngst abgerufener Objekte.
  - Conditional GET: Objekt wird nur übertragen, falls neuer als Cache-Version.
  - Caching proxy: Anfragen werden durch Proxy geleitet. Proxy verwaltet Cache ➔ gemeinsames Nutzen des Cache

Pipelining



Internet Relay Chat (IRC)

Internet Relay Chat

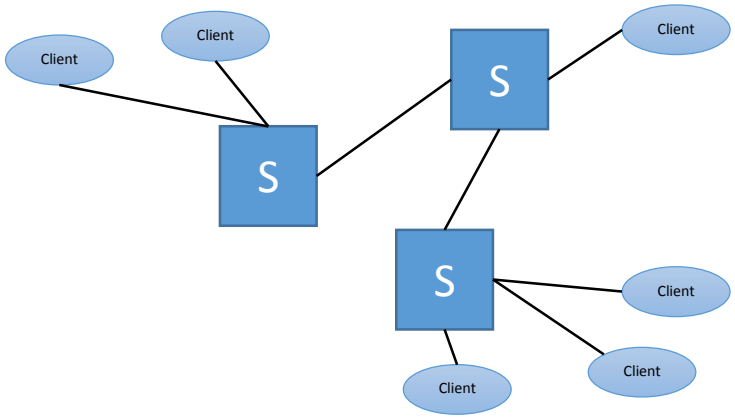
Der Chat in den Übungsaufgaben baut auf dem IRC-Protokoll auf (RFC 1459 https://tools.ietf.org/html/rfc1459).

Idee: jeder Client kann sich zu einem von mehreren Chatservern verbinden. Die Server leiten die Nachrichten der Clients untereinander weiter.

Nutzer können sich in Kanälen unterhalten oder in privaten Chats zu zweit. Beides funktioniert transparent über Servergrenzen hinweg.

IRC Anwendungsarchitektur

- Hybride Architektur
  - Client-Server-Architektur aus Sicht der Clients (Chatteilnehmer)
  - Mehrere Server sind (üblicherweise) in einer Baumstruktur miteinander verbunden
- Nachrichten werden nur in den benötigten Teilbaum weitergeleitet
- Jeder Server kennt alle Teilnehmer (Clients und Server)



IRC Nachrichtenformat in BNF

```
<message>  ::= [':' <prefix> <SPACE> ] <command> <params> <crlf>

<prefix>   ::= <servername> | <nick> [ '!' <user> ] [ '@' <host> ]

<command>  ::= <letter> { <letter> } | <number> <number> <number>

<SPACE>    ::= ' ' { ' ' }

<params>   ::= <SPACE> [ ':' <trailing> | <middle> <params> ]

<middle>   ::= <Any *non-empty* sequence of octets not including SPACE
or NUL or CR or LF, the first of which may not be ':'>

<trailing> ::= <Any, possibly *empty*, sequence of octets not including
NUL or CR or LF>

<crlf>     ::= CR LF
```

Die Syntax ist in der sog. Backus-Naur-Form (BNF) definiert. Damit und mit der ebenfalls im RFC beschriebenen Semantik kann jeder einen IRC-Client oder Server eigenständig implementieren. → Übung!

IRC Nachrichtenbeispiel

Aus Sicht von Bob:

Eine Nachricht von Alice trifft ein:

```
:alice!~irssi@irc1.mnm-team.org PRIVMSG bob :Hello, Bob
```

„Alice, verbunden mit dem Server irc1.mnm-team.org, sendet eine Nachricht an bob, nämlich Hello, world“

Bob antwortet via:

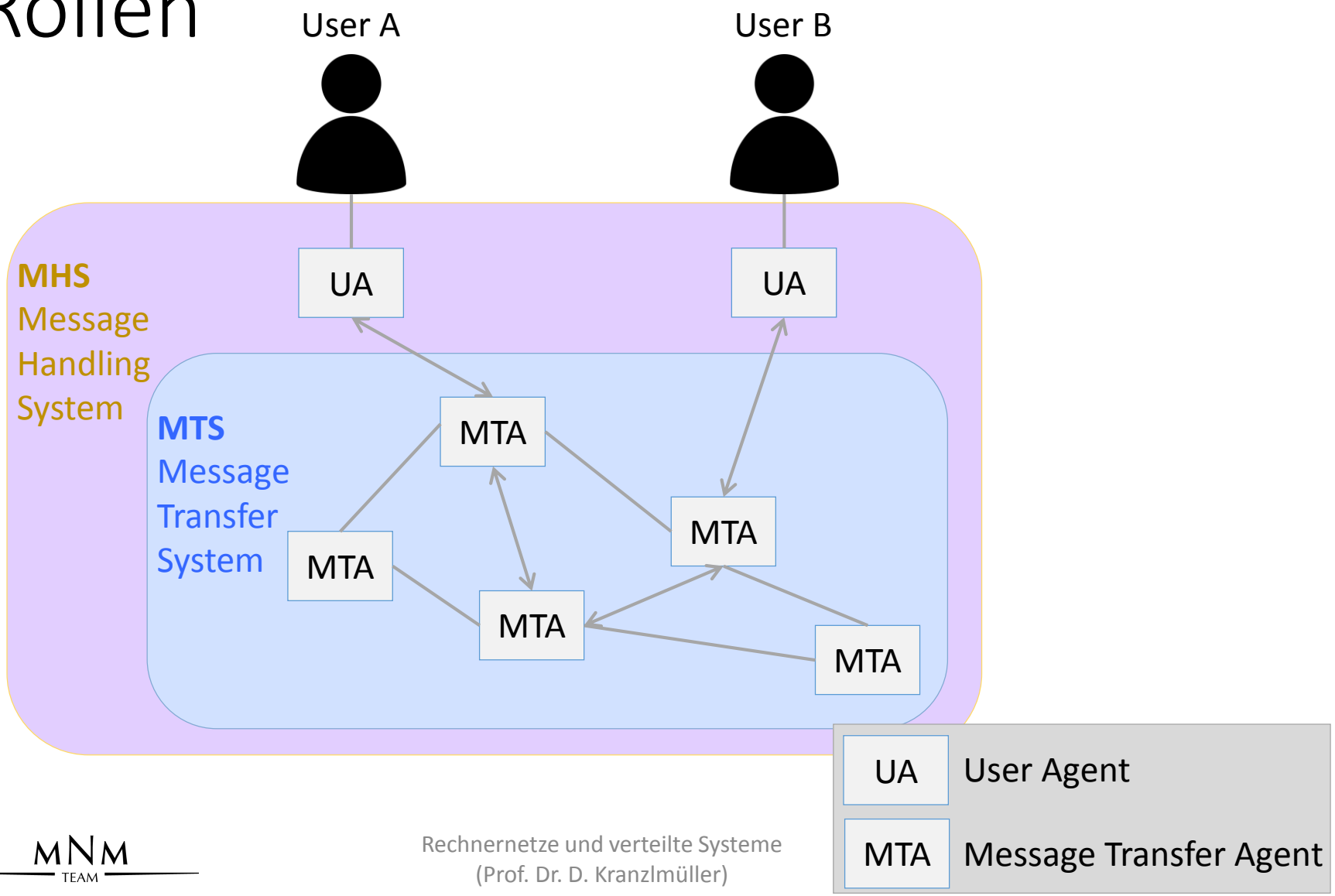
```
PRIVMSG alice :Hello, alice
```

Die weiteren Teile der Nachricht werden vom Server beim Weiterleiten hinzugefügt. Die Chatsoftware stellt diese Nachrichten dann in einem Chatfenster dar.

Electronic Mail (Email)

SMTP, MIME, POP, IMAP

Message Handling Systeme: Rollen



Protokolle

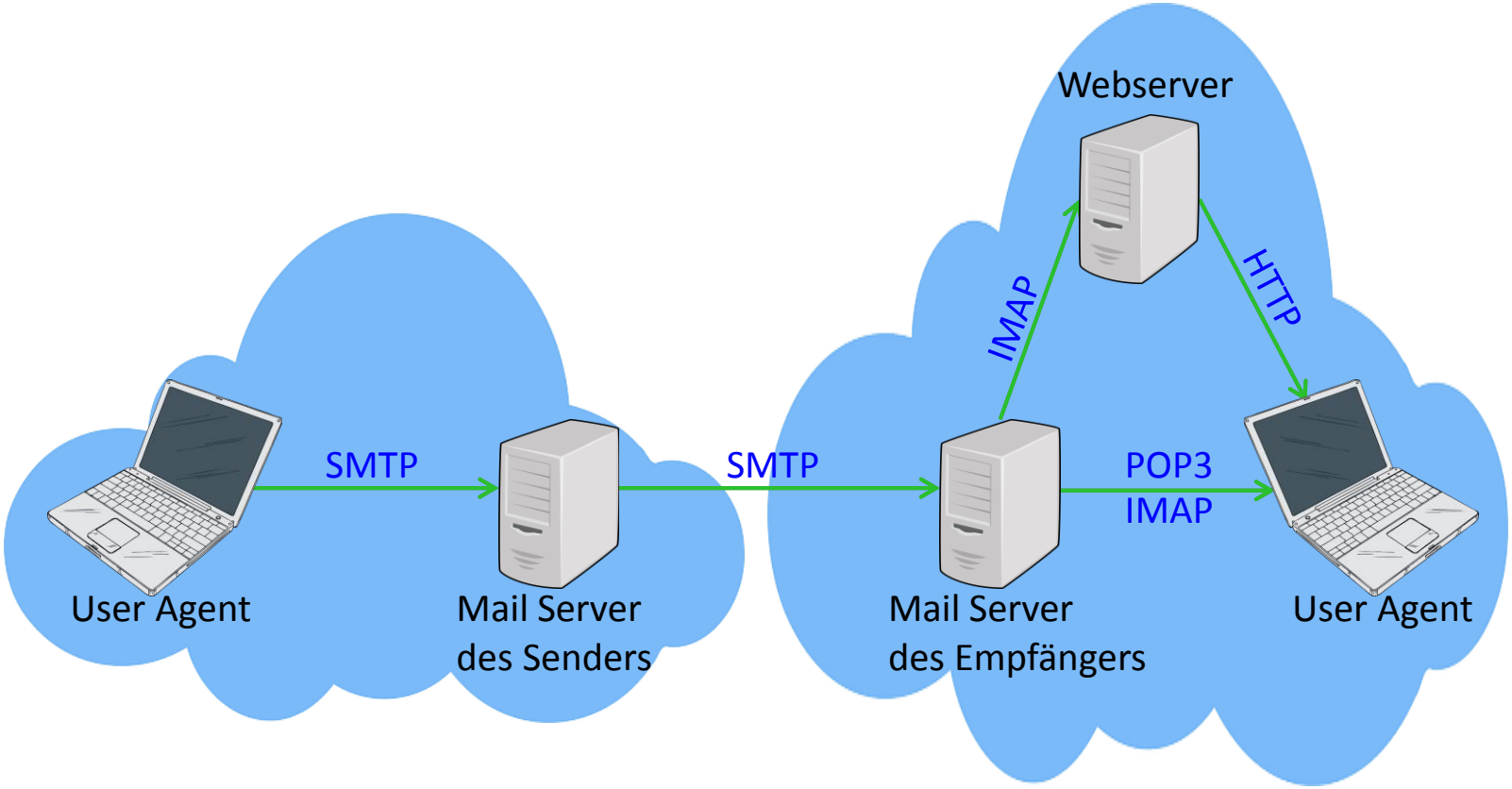
Senden von Emailnachrichten:

- SMTP (engl. Simple Mail Transfer Protocol)
  - in RFC 822 ausschließlich ASCII-Text
  - in RFC 2045/6 Erweiterung auf MIME (Binärdaten)

Abholen von Emailnachrichten:

- POP3 (engl. Post Office Protocol)
  - Authentifizierung/Autorisierung (User Agent, Server)
  - Übertragung der Nachrichten zum User Agent
- IMAP (engl. Internet Mail Access Protocol)
  - komplexer; Verwaltung mehrerer Ordner
  - Emailnachrichten können auf dem Server verwaltet werden
- HTTP (engl. Hypertext Transfer Protocol)
  - für browserbasierte Emaildienste

Protokolle (Animation)



Simple Mail Transfer Protocol (SMTP)

- Direkte, TCP-basierte (Port 25) Übertragung zwischen sendenden UA/MTA und empfangenden MTA
- spezifiziert in RFC 821
- Phasen: handshaking (greeting), transfer, closure
- Command/Response-Dialog (ASCII-basiert).
- Responses: Statuscode und Phrase.
- Relevanz der DNS-Informationen für Email
  - Problem: an welchen Host soll eine an nutzer@domain adressierte Email übertragen werden?
  - Resource Record des Typs **MX** nennt den für eine Domäne zuständigen Mailserver ➔ Emailadressen nicht notwendig an Namensraum für Rechner gebunden
  - Adressen wie vorname.nachname@unternehmen.com möglich

Beispiel: Einfache SMTP-Sitzung

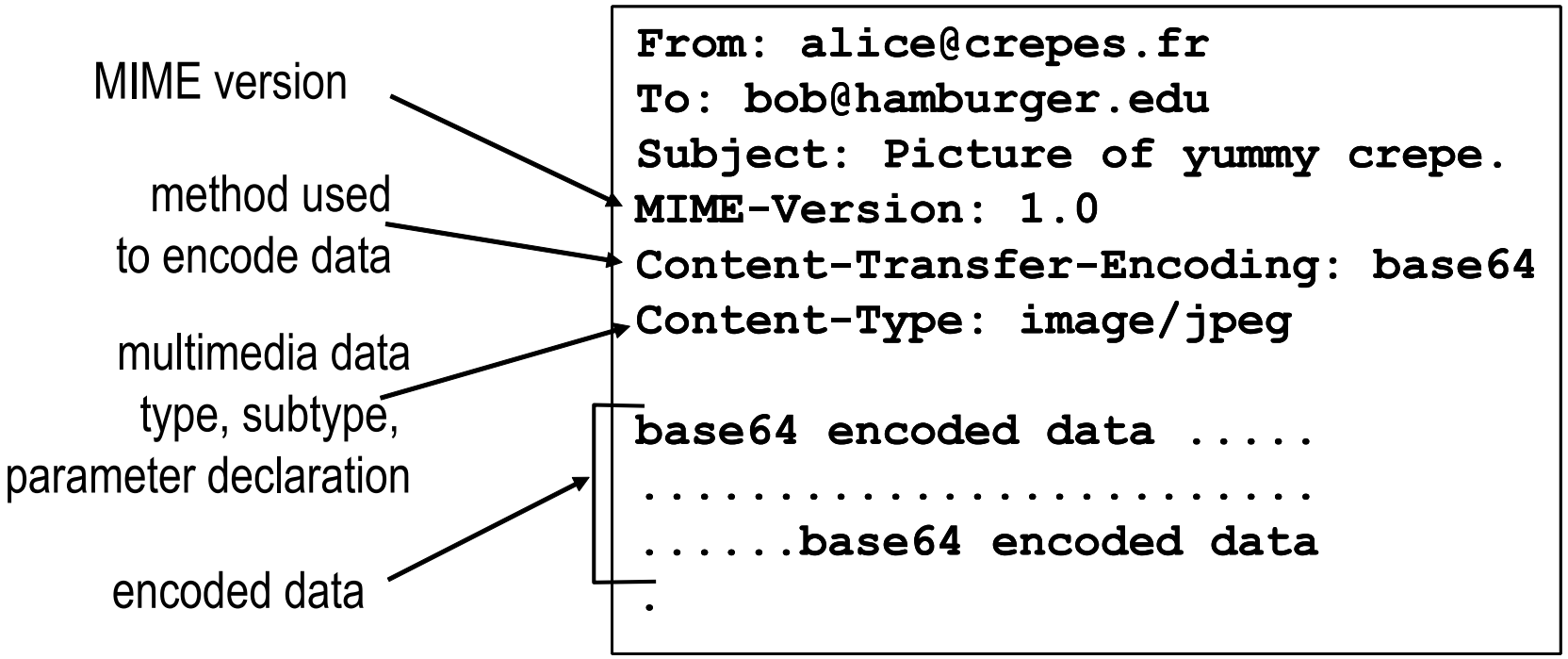
danciu@pcheger09: > **telnet mail 25** ←Server heißt „mail", Port ist 25
[. . .]
S: 220 mail.nm.ifi.lmu.de ESMTP Sendmail 8.12/Linux MNM 0.1; Mon, 28 Jan 2008 10:23:13 +0100 Unterstrichen: Statuscode
HELO nm.ifi.lmu.de Unterstrichen: Command
S: 250 mail.nm.ifi.lmu.de Hello pcheger09.in.nm.ifi.lmu.de, pleased to meet you
MAIL FROM:<danciu@nm.ifi.lmu.de>
S: 250 2.1.0 <danciu@nm.ifi.lmu.de>. . . Sender ok
RCPT TO:<rnp@nm.ifi.lmu.de>
S: 250 2.1.5 <rnp@nm.ifi.lmu.de>. . . Recipient ok
DATA
S: 354 Enter mail, end with "." on a line by itself
SMTP Probelauf. ←Text der Email
. ←Einsamer Punkt
S: 250 2.0.0 m0SJNDno027963 Message accepted for delivery
QUIT
S: 221 2.0.0 mail.nm.ifi.lmu.de closing connection
Connection closed by foreign host. ←Nachricht von telnet-Client

S:. . . :SMTP-Server **fett**: Benutzereingabe farbig: Kommentare

Nachrichtenformat

- Standard für Textnachrichten, spezifiziert in RFC 822
- Nachricht besteht aus Header, Body sowie Abschlusszeile
- Header-Zeilen (siehe auch Beispiel in Kapitel 3)
  - To, From, Subject
  - CC, BCC ([Blind] Carbon Copy)
  - Reply-To: Emailadresse, die für Antwort benutzt werden sollte
  - Message-Id: identifiziert eine Emailnachricht in späterer Kommunikation
  - In-Reply-To, References: Verweise auf Message-Ids von Emailnachrichten
  - Received: wird von jedem vermittelnden MTA dem Header hinzugefügt
- Body
  - Nutzdaten („Brief"); nur ASCII-Zeichen zulässig
  - Letzte Zeile markiert Nachrichtenende; sie enthält nur einen Punkt '.'
- Erweiterung zum Transport von Multimedia-Daten
  - MIME: multimedia mail extension, RFC 2045, 2056
  - zusätzliche Information im Header ➔ Angabe des MIME content type

Multi-purpose Internet Mail Extension (MIME)



Post Office Protocol (POP)

Autorisationsphase

- Client Befehle:
  - user: Benutzername
  - pass: Passwort
- Server Antworten: +OK, -ERR

Transaktionsphase

- list: list message numbers
- retr: retrieve message by number
- dele: delete message by number
- quit

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

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: quit
S: +OK POP3 server signing off

Internet Mail Access Protocol (IMAP)

- Problem: Mit POP3 können Nachrichten nur abgeholt, aber nicht auf dem Server verwaltet werden.
  ➔ Schlechte Unterstützung für nomadische Nutzer
- IMAP: Verwaltung von Emailnachrichten auf dem Server
  - Hierarchie von Ordnern (folders), die Nachrichten enthalten (wie Dateisystem)
  - Abrufen, Verschieben (zwischen Ordnern), Löschen von Nachrichten durch UA
  - Selektive Übertragung von Teilen von Emailnachrichten (z.B. nur Header)
- Last / Performanz
  - Verwaltungsoperationen auf Server ausgeführt ➔ belastet Servermaschine
  - Selektive Übertragung: Abruf nur relevanter Nachrichtenteile, z.B. über Verbindung mit niedriger Übertragungsrate

Kapitel 3.5
Protokolle und Standards

Allgemeines zu Protokollspezifikationen und Standards

Definition: Protokoll

Ein **Protokoll** (engl.: Protocol) ist eine Spezifikation (bzw. Vereinbarung) der Regeln, nach denen ein gegebener Informationsaustausch (eine gegebene Kommunikation) stattfindet. Insbesondere werden Syntax, Semantik und Synchronisationsverhalten einzelner *Nachrichten* festgelegt.

Protokollfunktionen

- Verbindungsmanagement
- Datenhandhabung
- Fehlerbehandlung

- Beschreibung in der Protokollspezifikation

Typische Protokollfunktionen (1/3)

- Verbindungsmanagement
  - Aufbau, Abbau
  - Multiplexing, Splitting
  - Protokoll Selektion (auf benachbarten Schichten)

Achtung:
Transporteinheiten haben schichtspezifisch verschiedene Bezeichnungen (z.B.: Nachricht, Segment, Paket, Block, Frame)

Typische Protokollfunktionen (2/3)

- Datenhandhabung
  - Zerlegung (Segmenting) und Zusammenfügung (*Reassembly)* von Dateneinheiten
  - Versehen von Nutzdaten mit Steuerinformationen
  - Wegewahl (*routing*)
  - Flussteuerung (*flow control*)

Typische Protokollfunktionen (3/3)

- Fehlerbehandlung
  - Verfälschung der Daten:
    Prüfsummen (CRC, BCC), Paritätsbits, ...
  - Verlust der Daten:
    Sequenznummern, Quittungen, Timeout
  - Duplikate:
    Sequenznummern
  - Falsche Reihenfolge:
    Sequenznummern
  - Verbindungsabbruch:
    Reset, Wiederaufsatzpunkte

Protokollspezifikation (1/2)

Aufgabe:

- Arbeitsteilung und Zusammenarbeit mit Protokollen benachbarter Schichten

Methoden der Protokollspezifikation:

- erweiterte endliche Automaten [ESTELLE]
- Ablaufdiagramme [SDL]
- Sequenzdiagramme
- Programmiersprachen [ASN.1]
- Temporale Logik [LOTOS]
- Petri-Netze (nach Carl Adam Petri)

Protokollspezifikation (2/2)

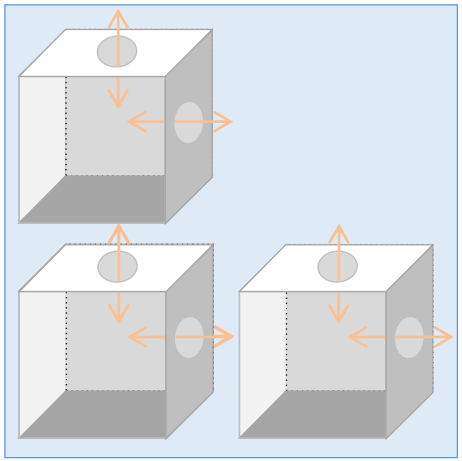
- Protokollverifikation und Validierung
  - Korrektheit
  - Verklemmungsfreiheit (Deadlock)

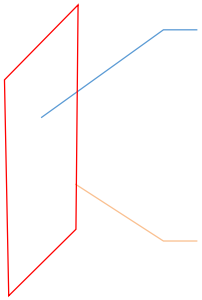
- Protokolltests
  - mit Partner:
    Interoperabilität (*interoperability)*
  - gegen Standard:
    Konformität/Übereinstimmung (*conformance)*

Kapitel 3.6 Schnittstellen und Dateneinheiten

Bestandteile der Schichtenarchitektur

Das Prinzip der Schnittbildung





Interaktionsstelle

Schnitt

- Systemschnitt
- Dienstschnitt
- Protokollschnitt

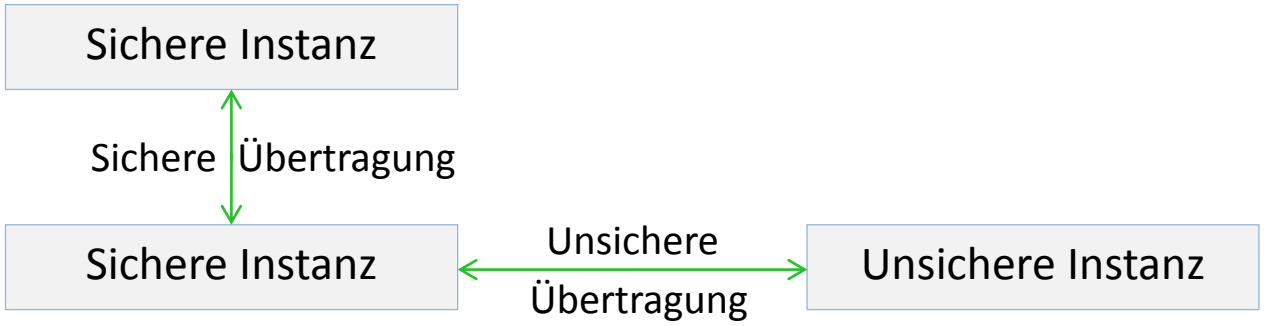
Schnitte dienen der ...

- Identifikation kommunizierender Einheiten und ihrer Interaktionsstellen
- Zuordnung von Interaktionsstellen zueinander
- Definition des inhaltlichen und zeitlichen Zusammenhangs zwischen Interaktionen

Analogie aus der OO-Welt

- Klasse / Objekt
- Gekapseltes Innenleben
- Schnittstelle nach aussen

Modellannahmen bei Schichtenarchitekturen



- **sicherer Kanal**: keine Fehler, kein Datenverlust, kein Synchronisationsverlust
- **sichere Instanz**: kein Fehlverhalten, kein Synchronisationsverlust
- **unsichere/r Instanz/Kanal**: sonst

Schnittbildung in der Schichtenarchitektur

- Systemschnitt
- Dienstschnitt
- Protokollschnitt

Systemschnitt

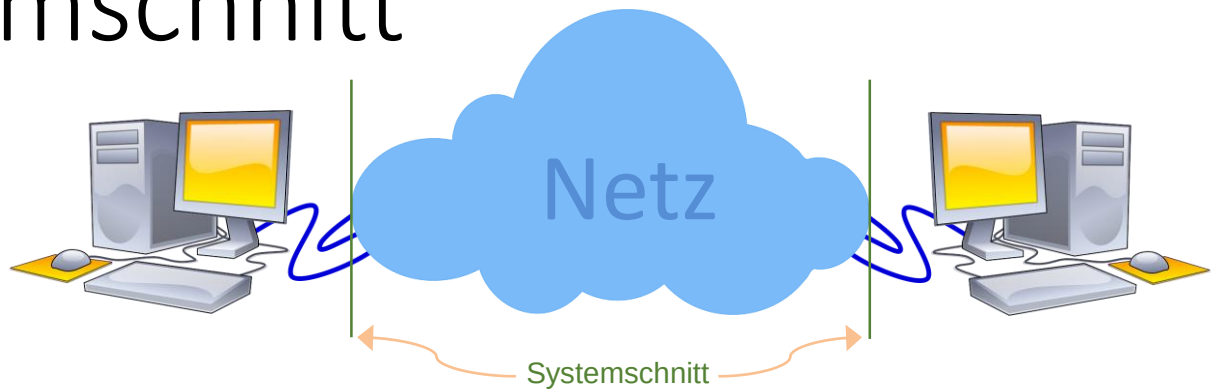
Systemschnitte

- setzen an den tatsächlichen (physikalischen) Grenzen der kommunizierenden Systeme an.

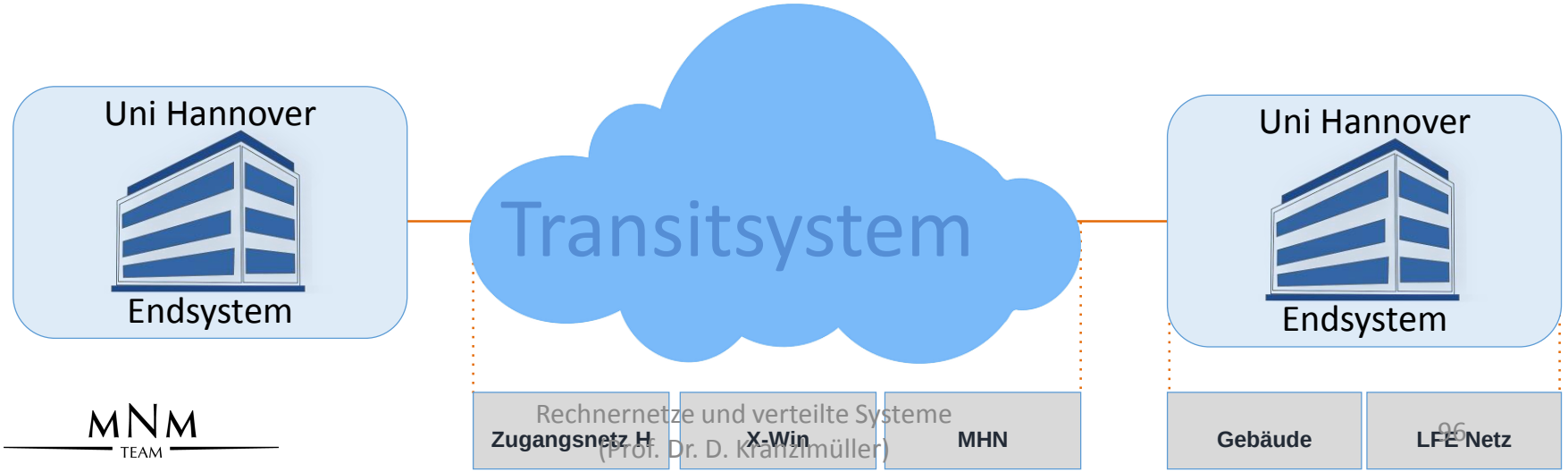
  Beispiel: Ein Laptop ist ein System, ein Router ist ein System und das Ethernet Kabel das sie verbindet ist die Schnittstelle zwischen diesen Systemen.

- dienen der Festlegung diskreter kommunizierender Systeme.
- können zur Verfeinerung (Annäherung an reale Konfiguration) mehrfach angewendet werden.

Systemschnitt



- Festlegung diskreter kommunizierender Systeme
- Interaktionsstelle stimmt mit dem realen Übergang zum Medium überein.
- Kann zur Verfeinerung (Annäherung an reale Konfiguration) mehrfach angewendet werden (z.B. unten im Bild).

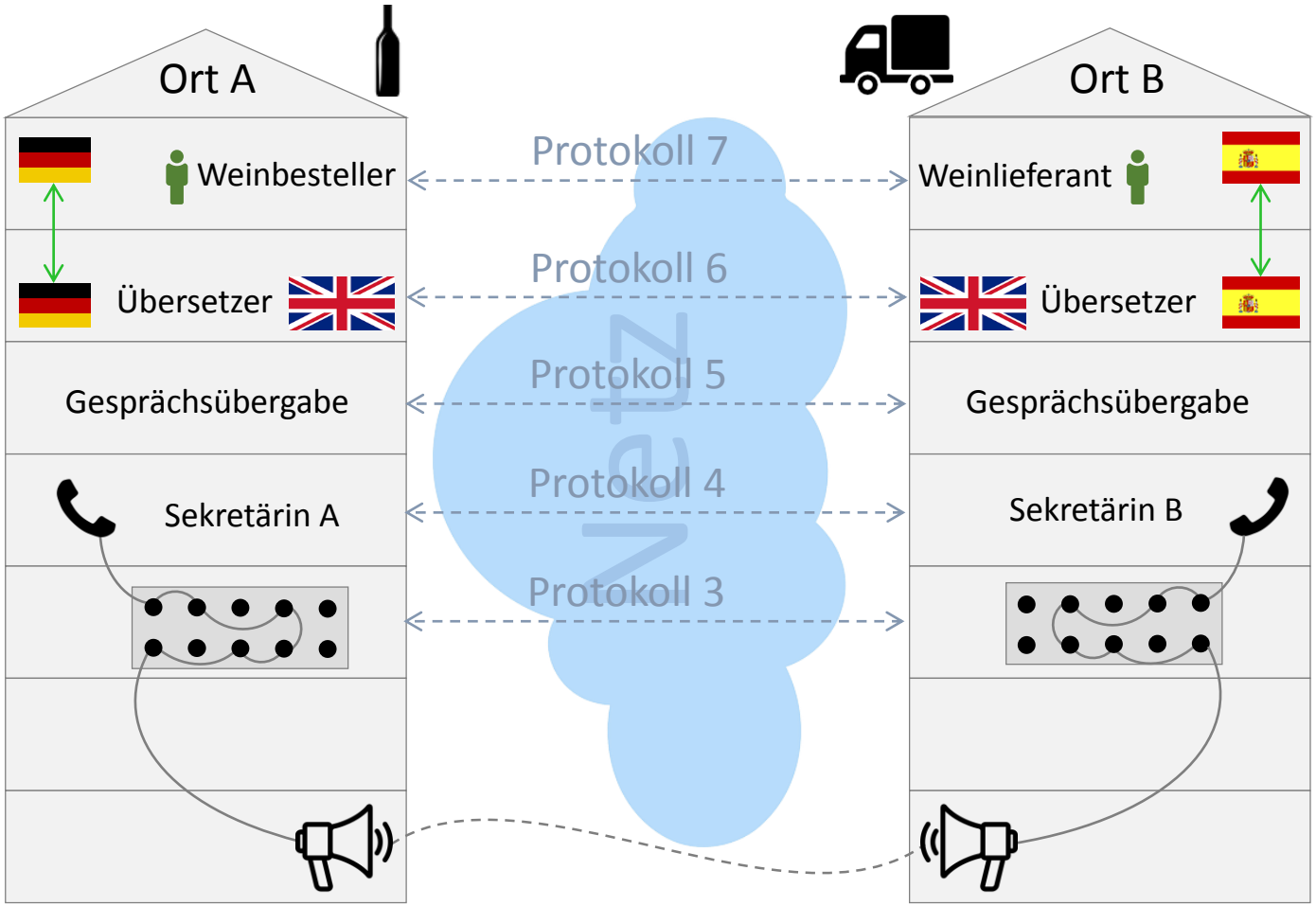


Dienstschnitt

Dienstschnitte

- setzen zwischen den Diensten zweier benachbarter Schichten an.
  - Der Kommunikationsvorgang innerhalb eines Systems wird in Teilvorgänge aufgeteilt.
- legen Dienstnutzer, Dienst, und Diensterbringer fest.
- ergeben nur in Verbindung mit einer Schichtarchitektur Sinn.

Dienstschichtung (Beispiel)

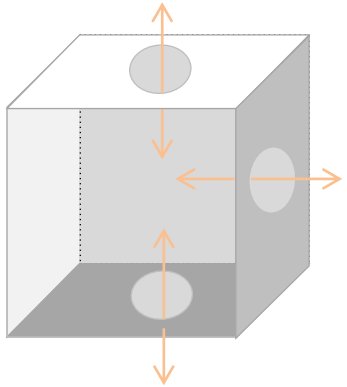


Protokollschnitt

- **Protokollschnitte** setzen zwischen den *Peer Entities* zweier kommunizierender Endsysteme an.
- *Peer Entities* sind Instanzen deselben Protokolls (auf der selben Schicht) in zwei verschiedenen kommunizierenden Systemen.
- Bsp.: Eine Instanz des TCP Protokolls auf unserem Computer kommuniziert mit einer Instanz des TCP Protokolls auf einem Webserver.
- Ziel: koordinierte Diensterbringung durch mehrere Systeme.

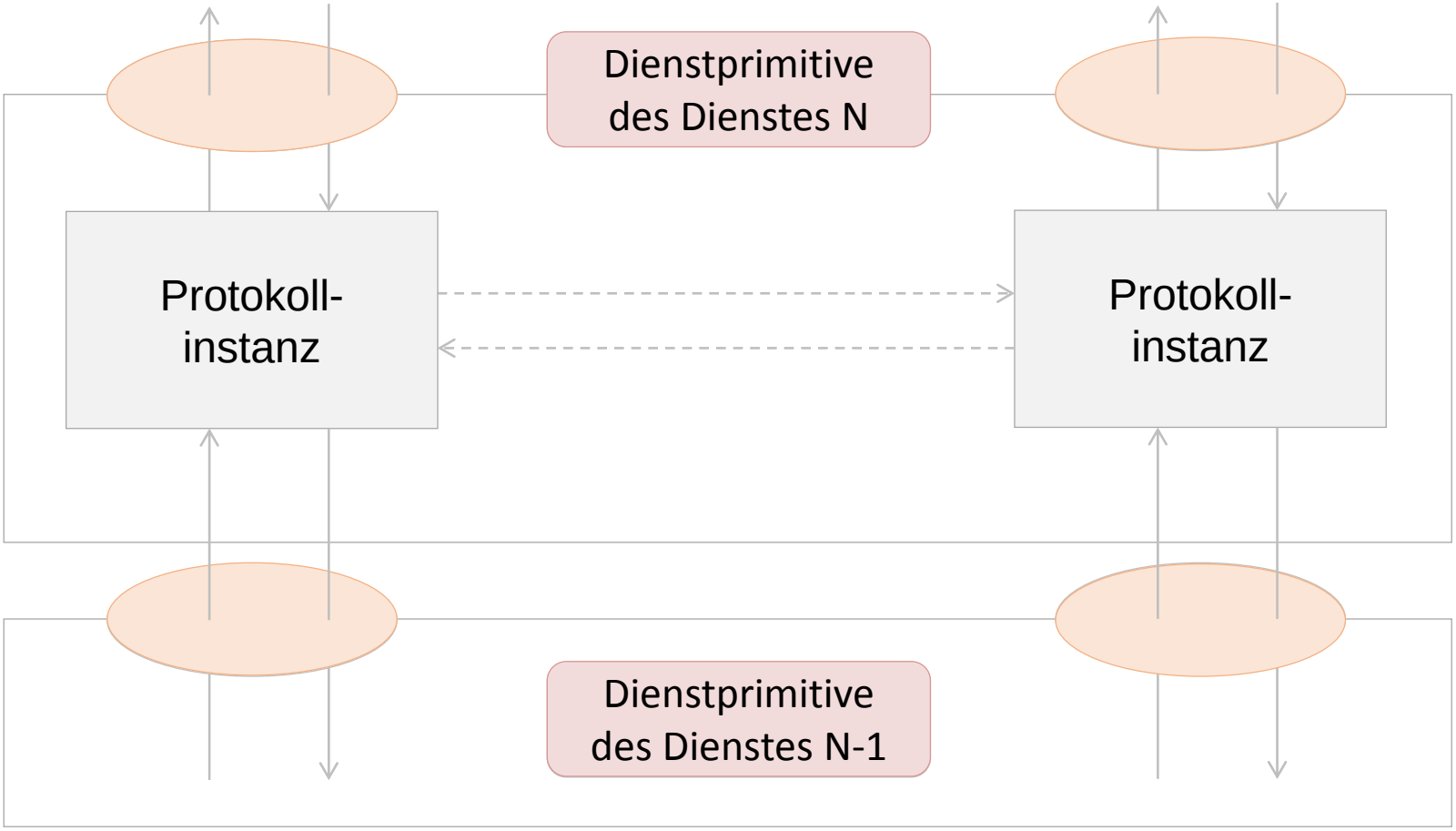
Protokollschnitt

- Protokollinstanzen haben drei Interaktionsstellen:
  - nach oben
  - nach unten
  - zur *Peer Entity*



- Dabei ist die Kommunikation der Protokollinstanzen (*Peer Entities*) **virtuell**.
- Tatsächlich kann nur nach oben oder unten kommuniziert werden.

Protokollinstanzen



Schnittbildung in der Schichtenarchitektur

- Systemschnitt – zwischen Systemen
- Dienstschnitt – zwischen benachbarten Schichten
- Protokollschnitt – zwischen Peer Entities

Dateneinheiten (1/4)

- Innerhalb des ISO/OSI-Modells (nach unten, nach oben, und horizontal) unterscheiden wir allgemein zwischen
  - *Nutzdaten*
  - *Steuerinformation*
- Welcher Teil **Steuerinformation** ist und welcher **Nutzdaten**, kommt darauf an
  - auf welcher Schicht (in welcher Protokollinstanz) wir uns befinden
  - ob wir die Kommunikation nach oben, nach unten, oder zur *Peer Entity* betrachten.

Dateneinheiten (2/4)

- Horizontale Kommunikation (mit der *Peer Entity*):
  - Steuerinformation „**PCI**“ (Protocol Control Information, deutsch: Protokoll-Steuer-Information).
  - Nutzdaten „**UD**“ (User Data, deutsch: Benutzerdaten)
  - Einheit aus PCI und UD ➔ „**PDU**“ (Protocol Data Unit, deutsch: Protokoll-Dateneinheit)

Dateneinheiten (3/4)

- Vertikale Kommunikation (nach unten, oder nach oben) mit einer Protokollinstanz einer benachbarten Schicht:
  - Steuerinformation „**ICI**“ (Interface Control Information, deutsch: Schnittstellen-Steuerdaten)
  - Nutzdaten „**ID**“ (Interface Data, deutsch: Schnittstellen-Daten)
  - Einheit aus ICI und ID ➔ „**IDU**“ (Interface Data Unit, deutsch: Schnittstellen-Dateneinheit)

Dateneinheiten (4/4)

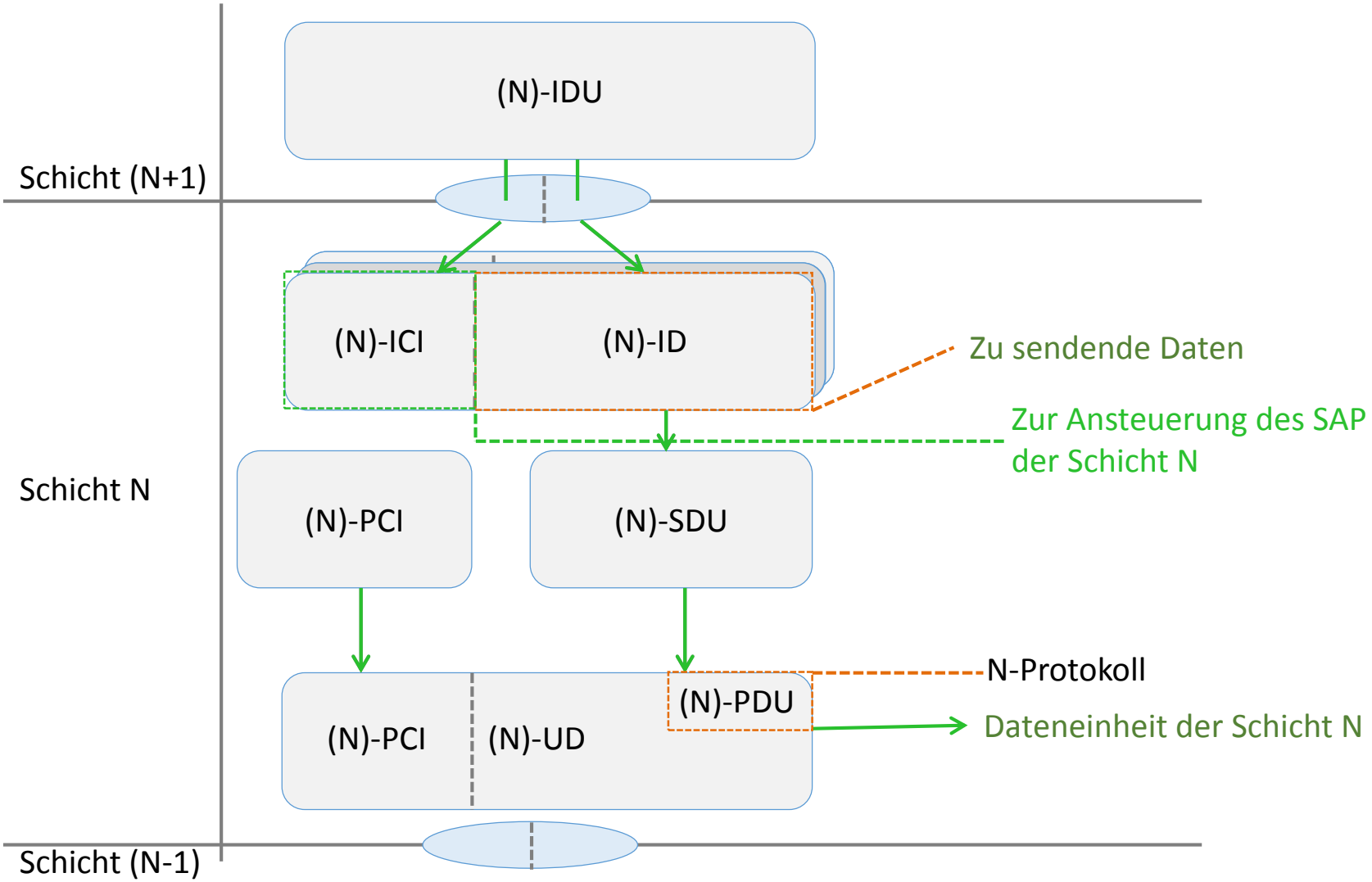
- SAP (Service Access Point): Interface/Schnittstelle
  - ICI (Interface Control Information) wird also zur Ansteuerung eines SAP verwendet.

- SDU (Service Data Unit): Wenn eine IDU die relevante Schnittstelle passiert hat, wird dessen ID auch als „SDU“ (Service Data Unit, deutsch: Dienst-Daten-Einheit) bezeichnet.

Zusammenfassung Dateneinheiten

Kommunikation	Steuerinformation	Nutzdaten	Kombiniert
Peer-Instanzen (horizontal)	Port-Steuer-Info (PCI)	Benutzer Daten (UD)	Protokoll Dateneinheit (PDU)
Benachbarte Instanzen (vertikal)	Interface Steuerdaten (ICI)	Interface Daten (ID)	Interface Dateneinheit (IDU)

- In der Praxis: Teile der PCI dienen auch als ICI, sodass die ICI nicht separat hinzugefügt werden muss.



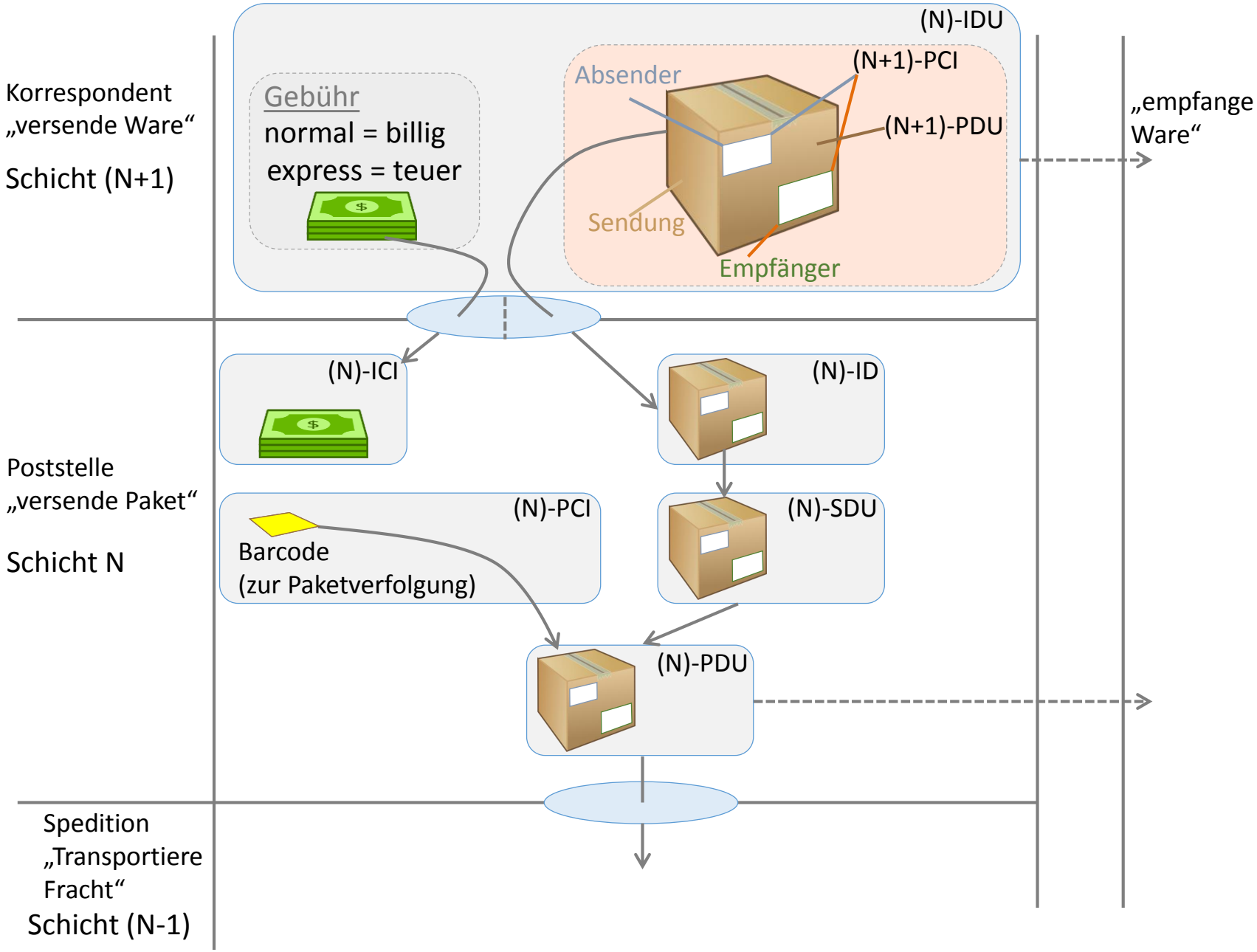
IDU: Interface Data Unit
ICI: Interface Control Information
PCI: Protocol Control Information

SDU: Service Data Unit
UD: User Data
PDU: Protocol Data Unit

MNM TEAM

Ablauf des Datenflusses

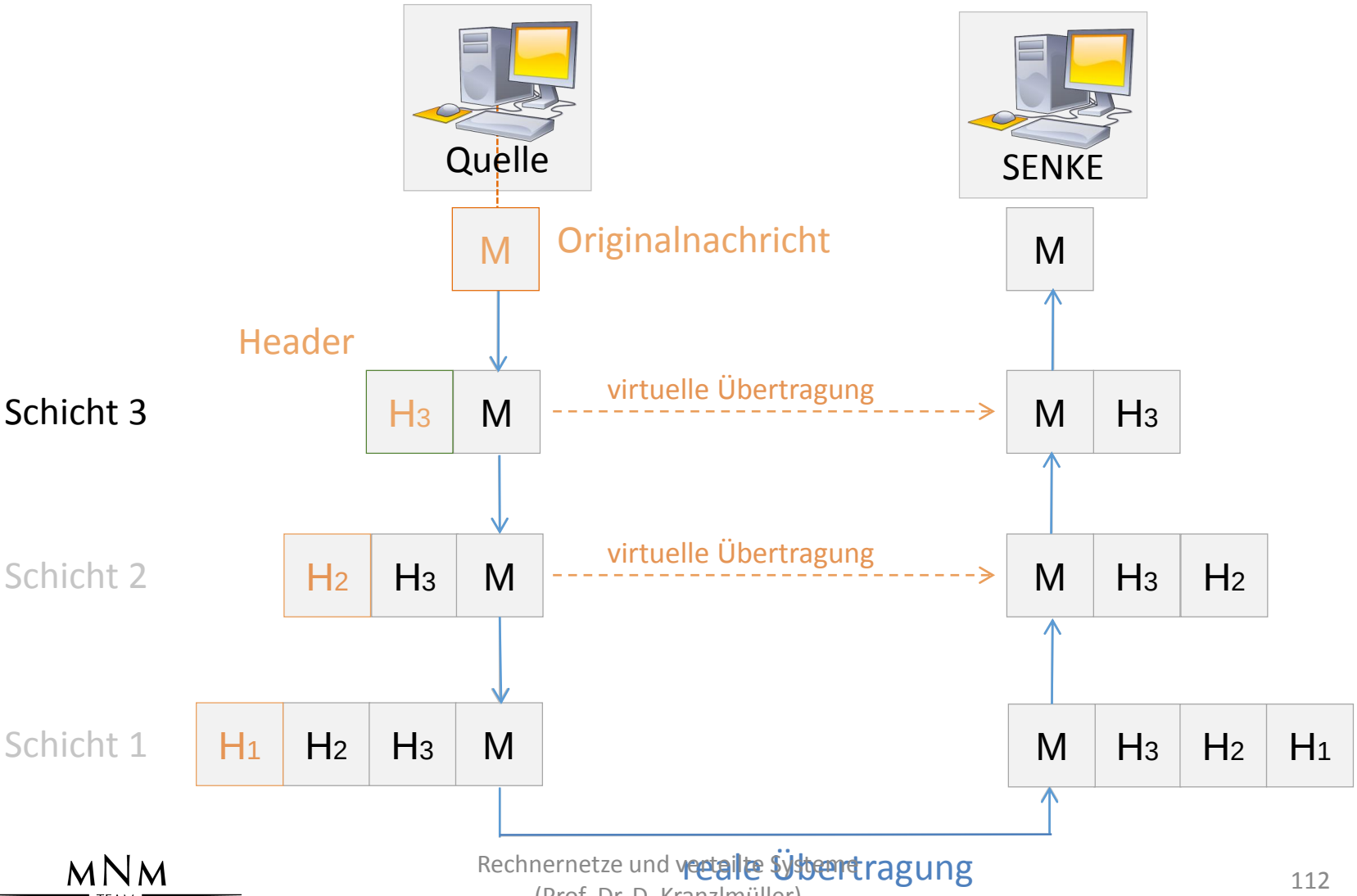
- (N)-IDU steuert SAP der Schicht N
  - (N)-ID = (N+1)-PDU
- Nach Passieren des SAP:
  - (N)-ID des (N)-IDU wird zu (N)-SDU
- (N)-SDU entspricht (N)-UD
- (N)-PDU ist (N)-PCI + (N)-UD



MNM TEAM

Analogie: **Paketversand**

Datenfluss (vertikal und horizontal)

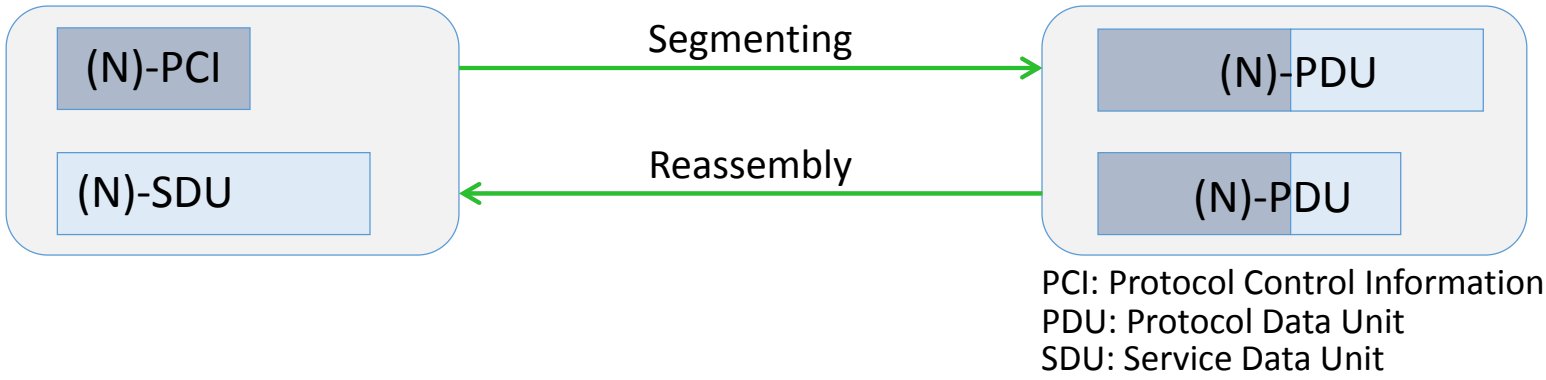


Abbildungen zw. Datenblöcken

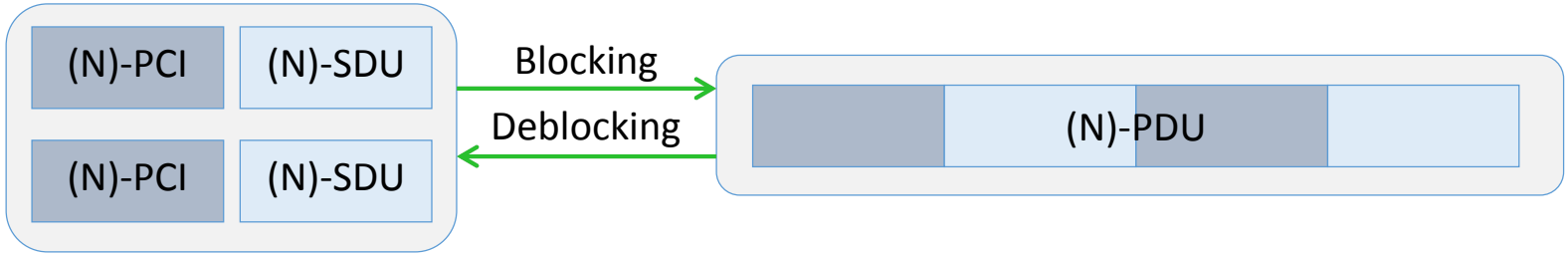
Operationen auf Datenblöcken:

- Segmentierung (Segmenting) / Neumontage (Reassembly)
- Blockieren (Blocking) / Deblockieren (Deblocking)
- Verkettung (Concatenation) / Trennung (Separation)

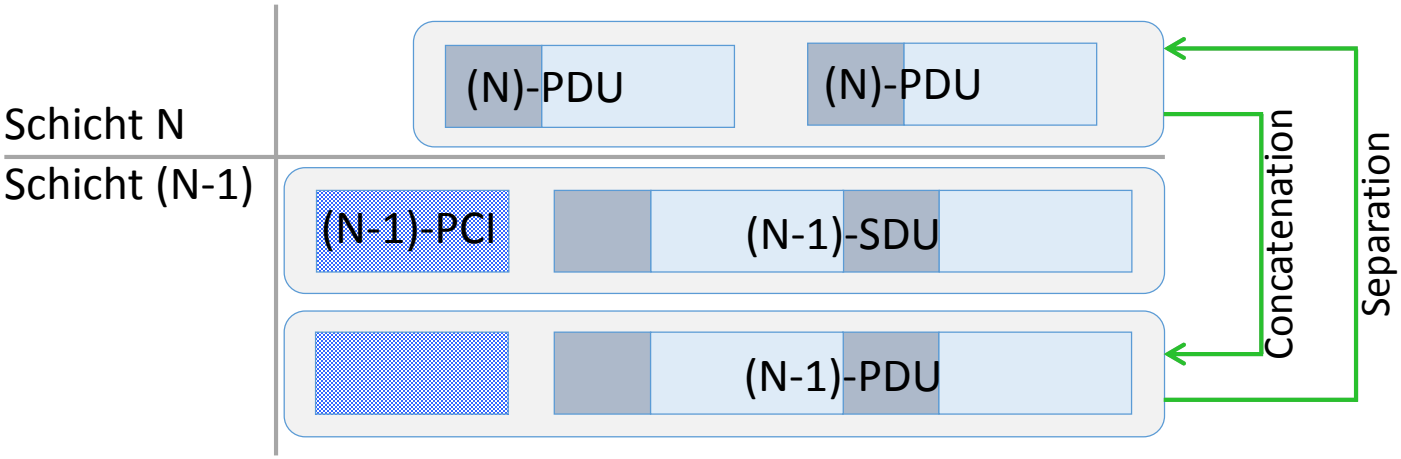
Abbildungen zw. Datenblöcken (Segmenting/Reassembly)



Abbildungen zw. Datenblöcken (Blocking/Deblocking)



Abbildungen zw. Datenblöcken (Concatenation&Separation)

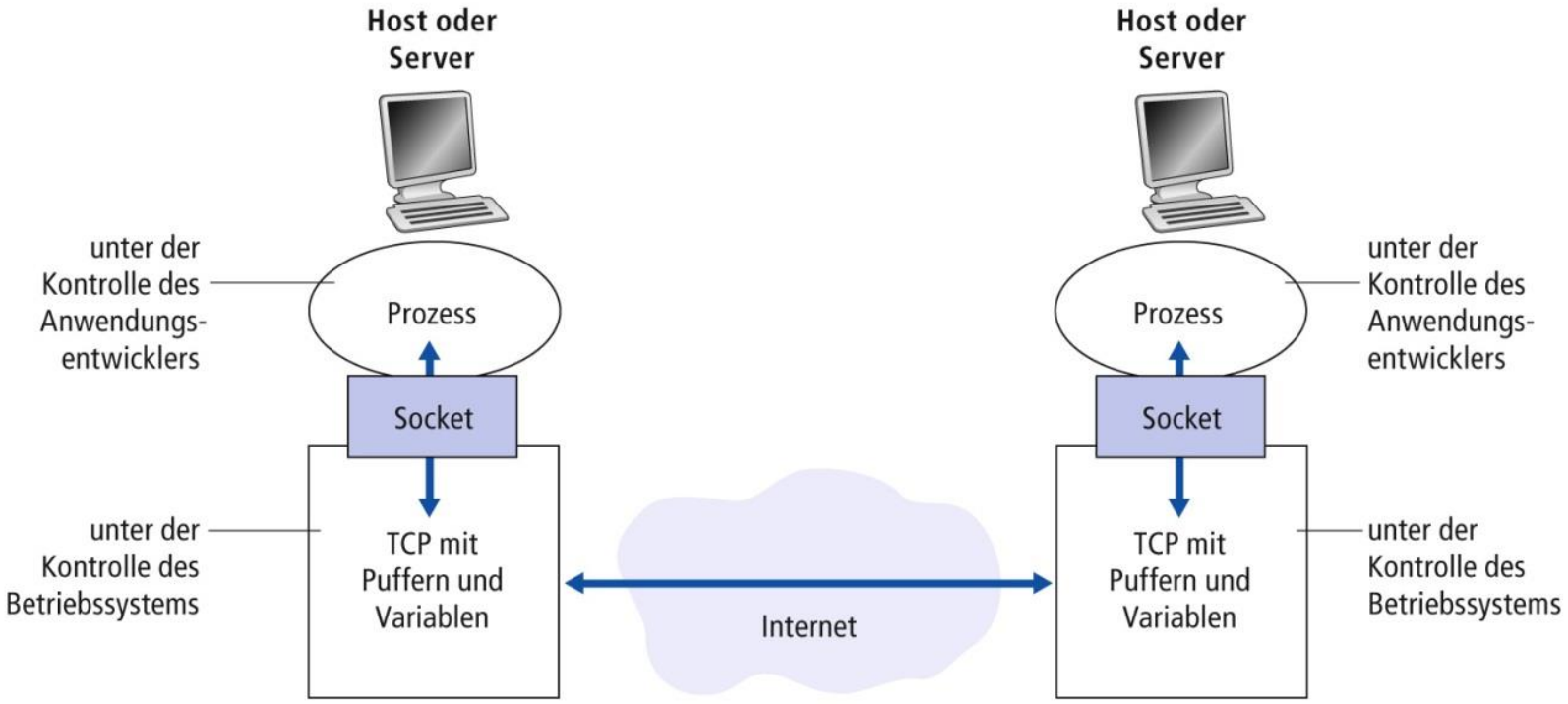


Kapitel 3.7
Ports und Sockets

Multiplexen über Anwendungen

Sockets

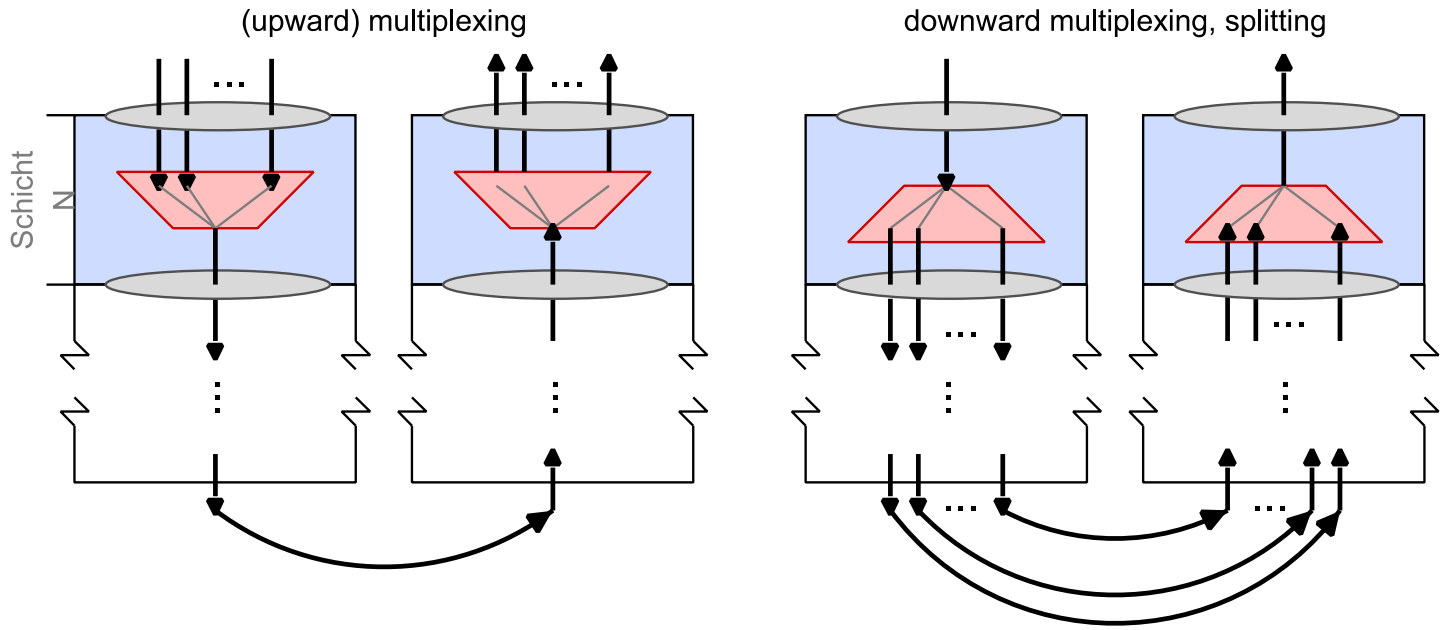
Sockets dienen als **Schnittstelle** zwischen Anwendungen und der Transportschicht im Internetmodell



Begriffsklärung

- **Socket**: Tupel bestehend aus IP-Adresse und Port.
  - Socket ist netzglobaler Kommunikationsendpunkt
  - vom Betriebssystem bereitgestellt
  - dienen als die **Schnittstelle** für Zugriff auf Transportdienst
- **Port**:
  - Adresse für Kommunikationsendpunkt einer Schicht 4 Protokollinstanz (auf gegebenem Host)
  - Flacher Adressraum (16-Bit Binärzahl)
- **Verbindung** besteht zwischen zwei Sockets

Der Multiplex Begriff

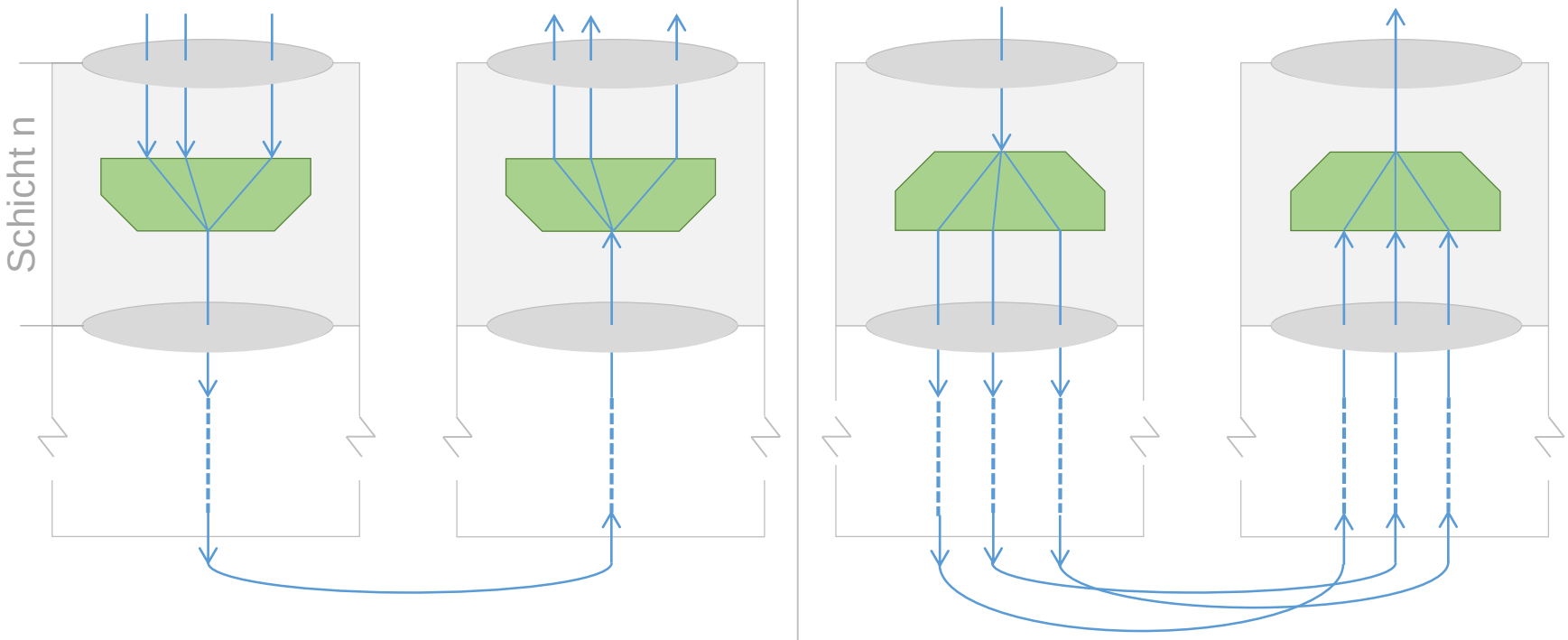


- upward multiplexing: Viele Kanäle einer höheren benachbarten Schicht werden zu einem Kanal einer niedrigeren benachbarten Schicht vereint.
- downward multiplexing: Ein Kanal einer höheren benachbarten Schicht wird in viele Kanäle einer niedrigeren benachbarten Schicht aufgeteilt.

Multiplexing (oft: „Mux“)

(upward) multiplexing

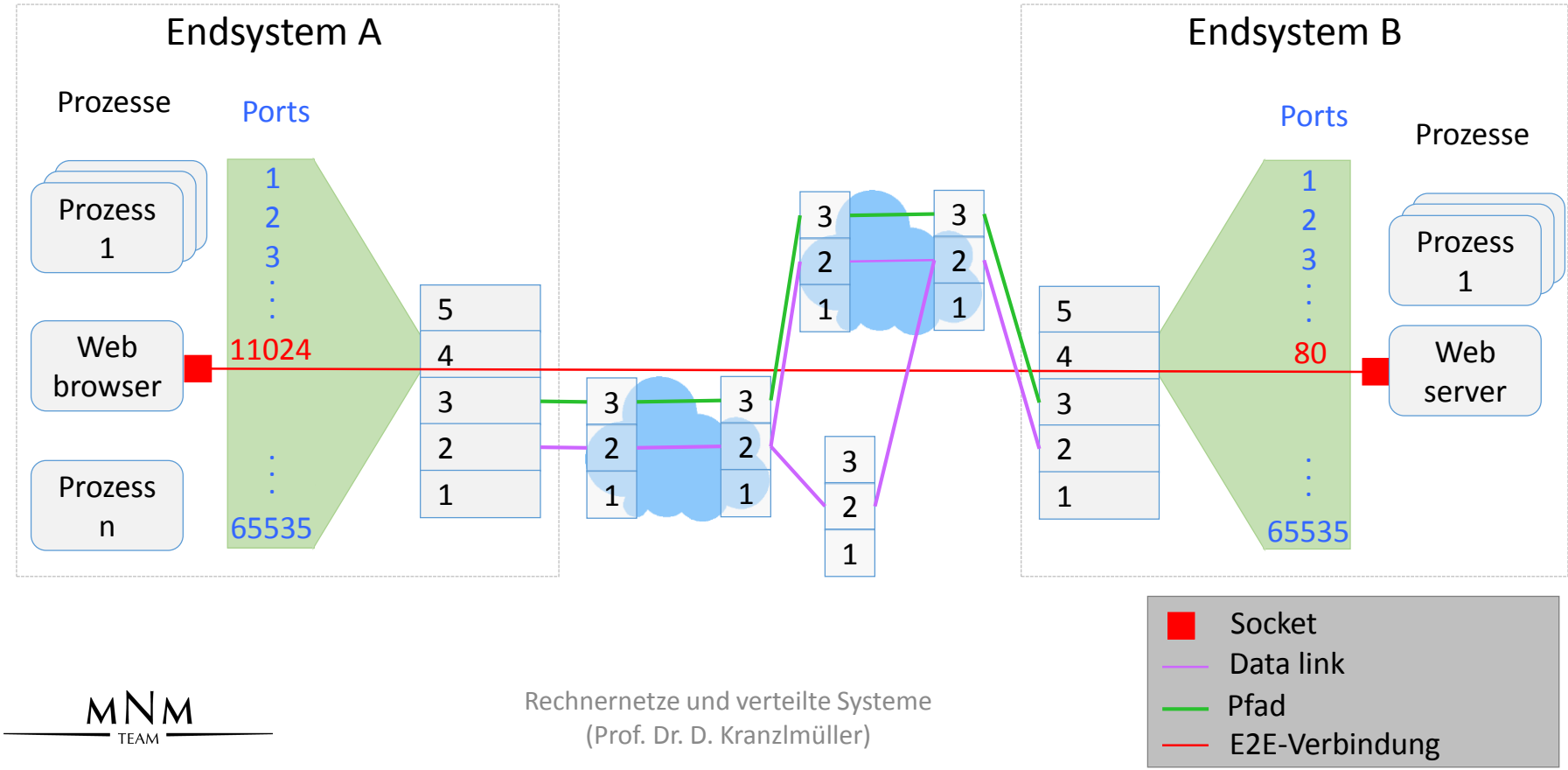
downward multiplexing, splitting



Multiplexen auf der Transportschicht im Internet

- Auf den Endsystemen im Internet gibt es i.d.R. viele Anwendungsprozesse, die über eine einzige Netzschnittstelle (IP-Adresse) Verbindungen aufbauen.
- Dies funktioniert, indem jeder solche Anwendungsprozess eigene Sockets mit einem eigenen Port verwendet.
- Über die Ports der Transportschicht wird also ein (upward) Multiplex über Anwendungsprozesse realisiert.

Programmierung von Sockets

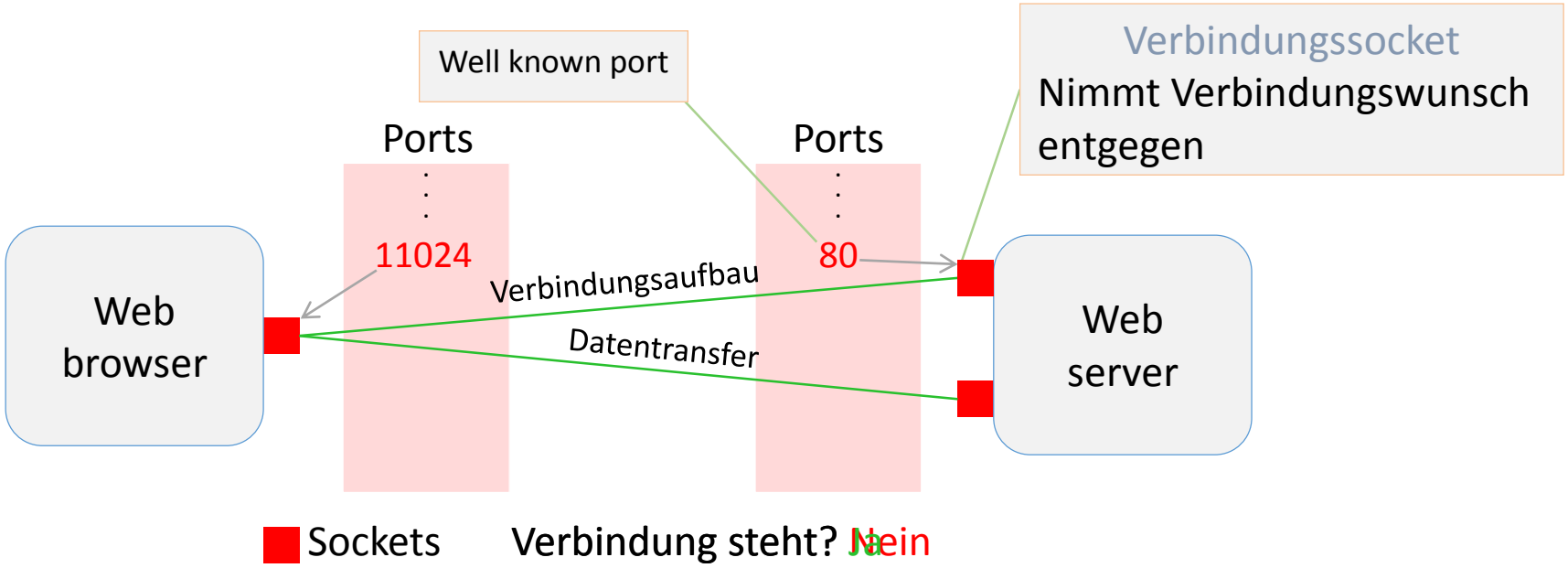


Sockets API

- Spezifikation in BSD 4.1 UNIX (1981)
- Umgang mit Sockets ähnlich wie mit einer Datei
  - „Everything is a file in UNIX“ [1]
  - Socket wird erzeugt/geöffnet
  - Es wird zum Socket geschrieben / vom Socket gelesen
  - Socket wird geschlossen/zerstört
- Client-Server-Paradigma
  - Server Socket: wartet auf Verbindungsanfragen von Clients
  - Normales Socket: wird nach dem Verbindungsaufbau von Client- sowie Serverprozessen zur Kommunikation genutzt.
  - Server/Daemonen: erzeugen neue Threads/Prozesse zur Abwicklung der Kommunikation nach dem Verbindungsaufbau.

[1] https://de.wikipedia.org/wiki/Everything_is_a_file

Beispiel: Verbindungssocket



TCP-Socket-Programmierungsbeispiel