Übungen zu Einführung in Rechnernetze
2. Übung

Tim Gerhard
[tim.gerhard]@kit.edu

Institut für Telematik, Prof. Zitterbart

Einführung in Rechnernetze – Übungsblatt 2

1. HTTP-Quiz
2. DNS in der Praxis
3. CDN, DNS, und DASH
4. SMTP in der Praxis
5. Unterschiede im Nachrichtenaustausch
6. Telnet und NetCat
7. Socket-Tutorial

HTTP-Quiz

Ein Nutzer fragt nach einer Webseite, die aus Text und drei Bildern besteht. Der Nutzer sendet eine Request-Nachricht und erhält darauf vier Response-Nachrichten.

Falsch!

- Es wird eine Response-Nachricht geschickt, und der Nutzer muss für jedes Bild eine eigene Request-Nachricht senden.

HTTP-Quiz

Zwei verschiedene Webseiten (z.B. http://telematics.tm.kit.edu/staff.php und http://telematics.tm.kit.edu/ss2017_pse.php) können über dieselbe persistente Verbindung gesendet werden.

Richtig!

- Dafür sind persistente Verbindungen gedacht!
- Der Webserver entscheidet nicht über semantische Verbindungen von Objekten.

HTTP-Quiz

Mit nicht-persistenten Verbindungen können mehrere Objekte zwischen einem Webbrowser und einem Webserver über dieselbe TCP-Verbindung gesendet werden.

Falsch!

- Dafür sind *persistente* Verbindungen gedacht!

HTTP-Quiz

Das Date:-Header-Feld in einer HTTP-Response-Nachricht zeigt an, wann die Datei zuletzt geändert wurde.

Falsch!

- Das Date-Header-Feld gibt den Zeitpunkt der Anfrage an.

HTTP-Quiz

HTTP-Response-Nachrichten können keinen leeren Body haben.

Falsch!

- HEAD-Requests haben immer einen leeren Body.

Einführung in Rechnernetze – Übungsblatt 2

1. HTTP-Quiz
2. DNS in der Praxis
3. CDN, DNS, und DASH
4. SMTP in der Praxis
5. Unterschiede im Nachrichtenaustausch
6. Telnet und NetCat
7. Socket-Tutorial

(a) Ermitteln des lokalen Name-Server

- nmcli dev show | grep DNS

(b) IP-Adresse von www.kit.edu

- IPv4: dig www.kit.edu A

- IPv6: dig www.kit.edu AAAA

(b) Interpretation der dig-Ausgabe

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> www.kit.edu a
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36139
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;www.kit.edu.                           IN      A

;; ANSWER SECTION:
www.kit.edu.                69          IN      A       129.13.40.10

;; Query time: 2 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Thu May 11 10:58:56 CEST 2017
;; MSG SIZE  rcvd: 56
```

(c) Name-Server für uka.de

- dig uka.de NS

(d) student.kit.edu Mailserver

- dig student.kit.edu MX

(e)

- Die IP-Adresse des k-Root-Servers in Karlsruhe lautet: 193.0.14.129

- dig @193.0.14.129 edu. NS
  - ergibt 192.5.6.30 als möglichen Name-Server für .edu

- dig @192.5.6.30 kit.edu NS
  - ergibt 141.52.27.35 als möglichen Name-Server für kit.edu

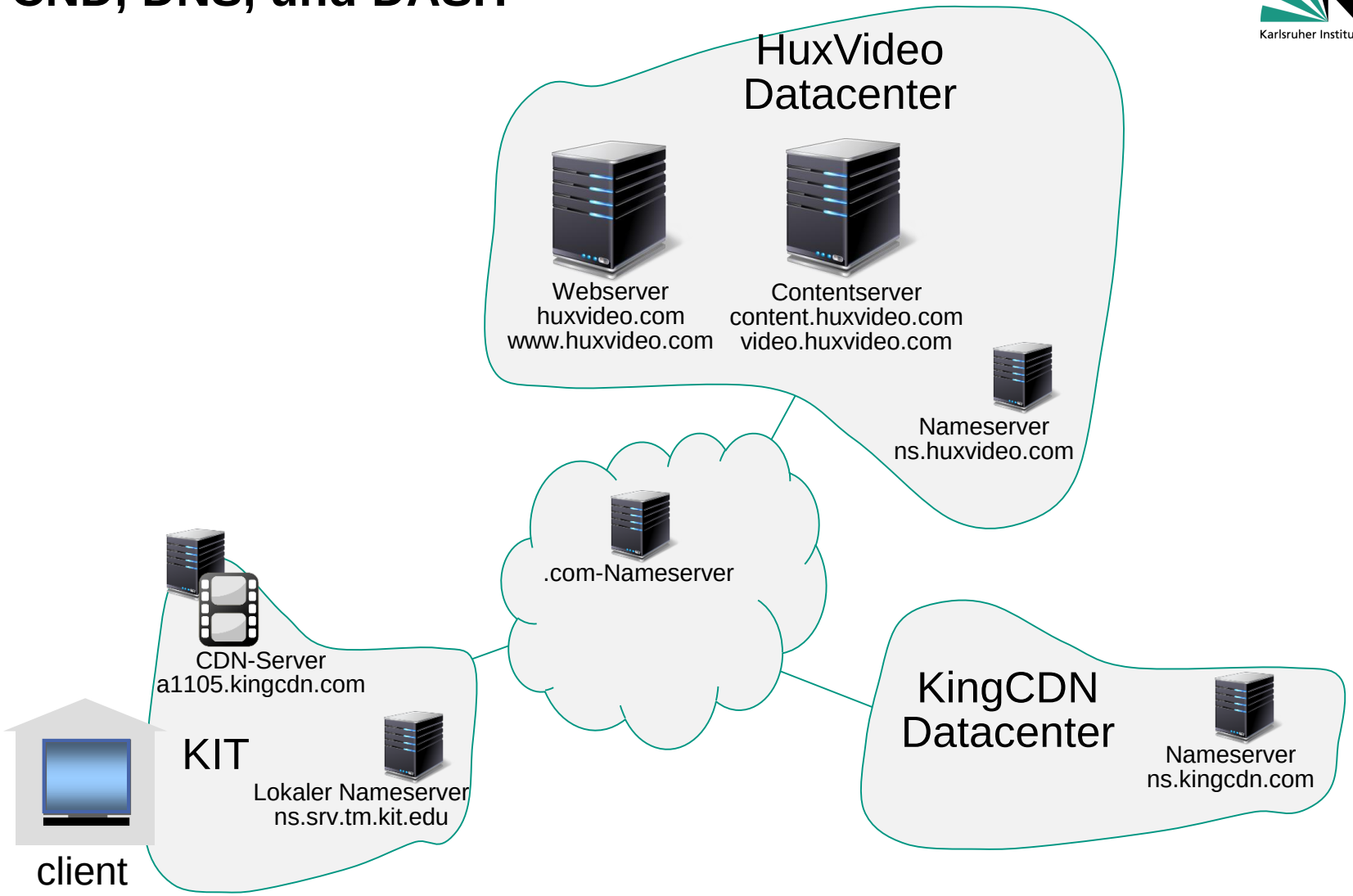
- dig @141.52.27.35 tm.kit.edu NS
  - ergibt den gleichen Name-Server

- dig @141.52.27.35 telematics.tm.kit.edu CNAME
  - gibt uns einen alternativen Hostname: telematics-atis-tm.scc.kit.edu
  - für diesen sind die gleichen Name-Server zuständig.
- dig @141.52.27.35 telematics-atis-tm.scc.kit.edu A
  - gibt uns dann die Ziel-Adresse: 129.13.41.63

Einführung in Rechnernetze – Übungsblatt 2

1. HTTP-Quiz
2. DNS in der Praxis
3. CDN, DNS, und DASH
4. SMTP in der Praxis
5. Unterschiede im Nachrichtenaustausch
6. Telnet und NetCat
7. Socket-Tutorial

CND, DNS, und DASH



(a) Konfiguration des Name-Servers

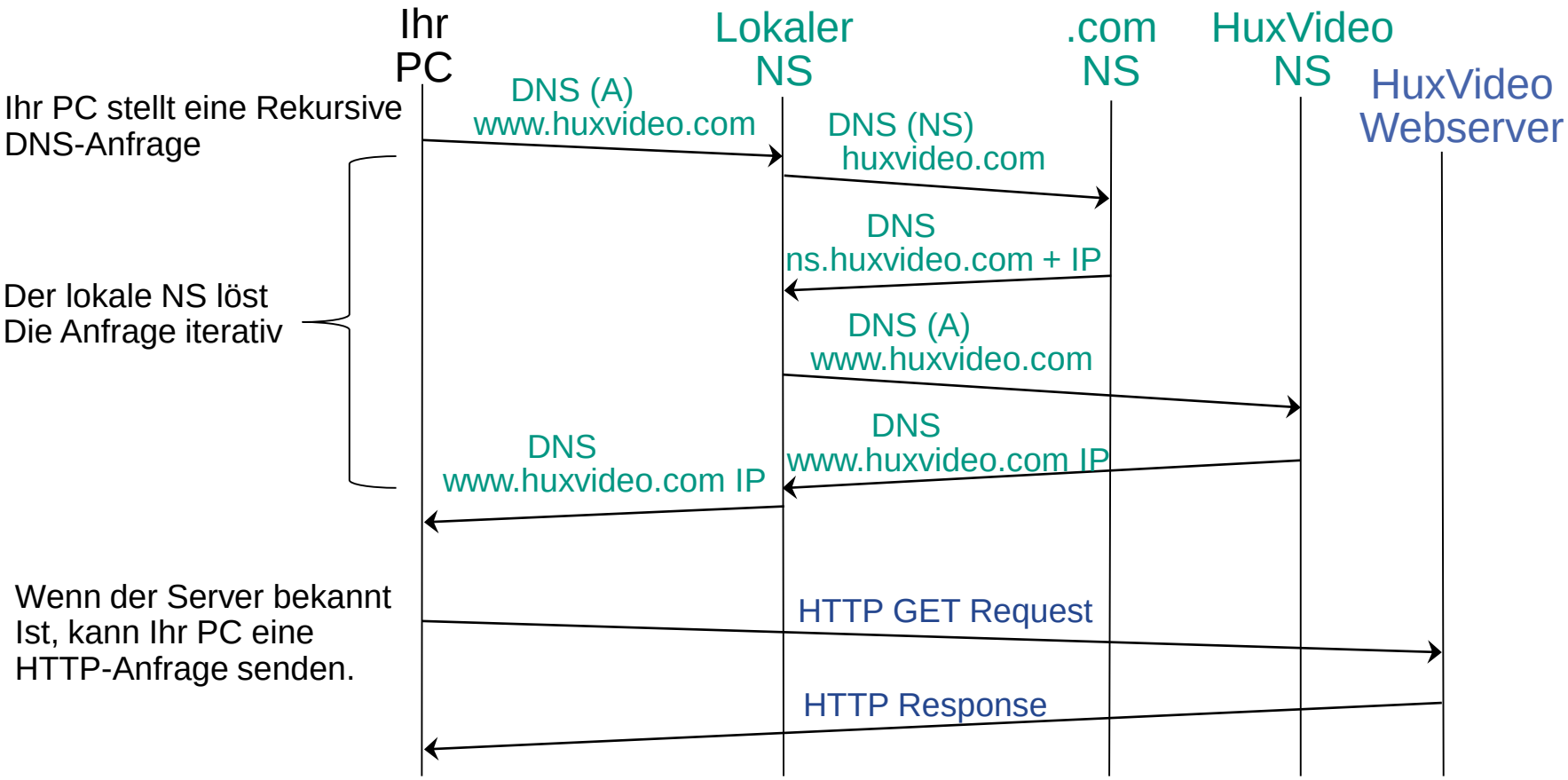
- huxvideo.com
  - A, AAAA (sofern gewünscht)
  - NS (ns.huxvideo.com)
  - MX (huxvideo.com) – (sofern gewünscht)
- www.huxvideo.com
  - A, AAAA
- content.huxvideo.com
  - A, AAAA
- video.huxvideo.com
  - CNAME oder A,AAAA
- ns.huxvideo.com
  - A, AAAA

(b) Registrierung der Domain

- IP-Adresse des Name-Servers

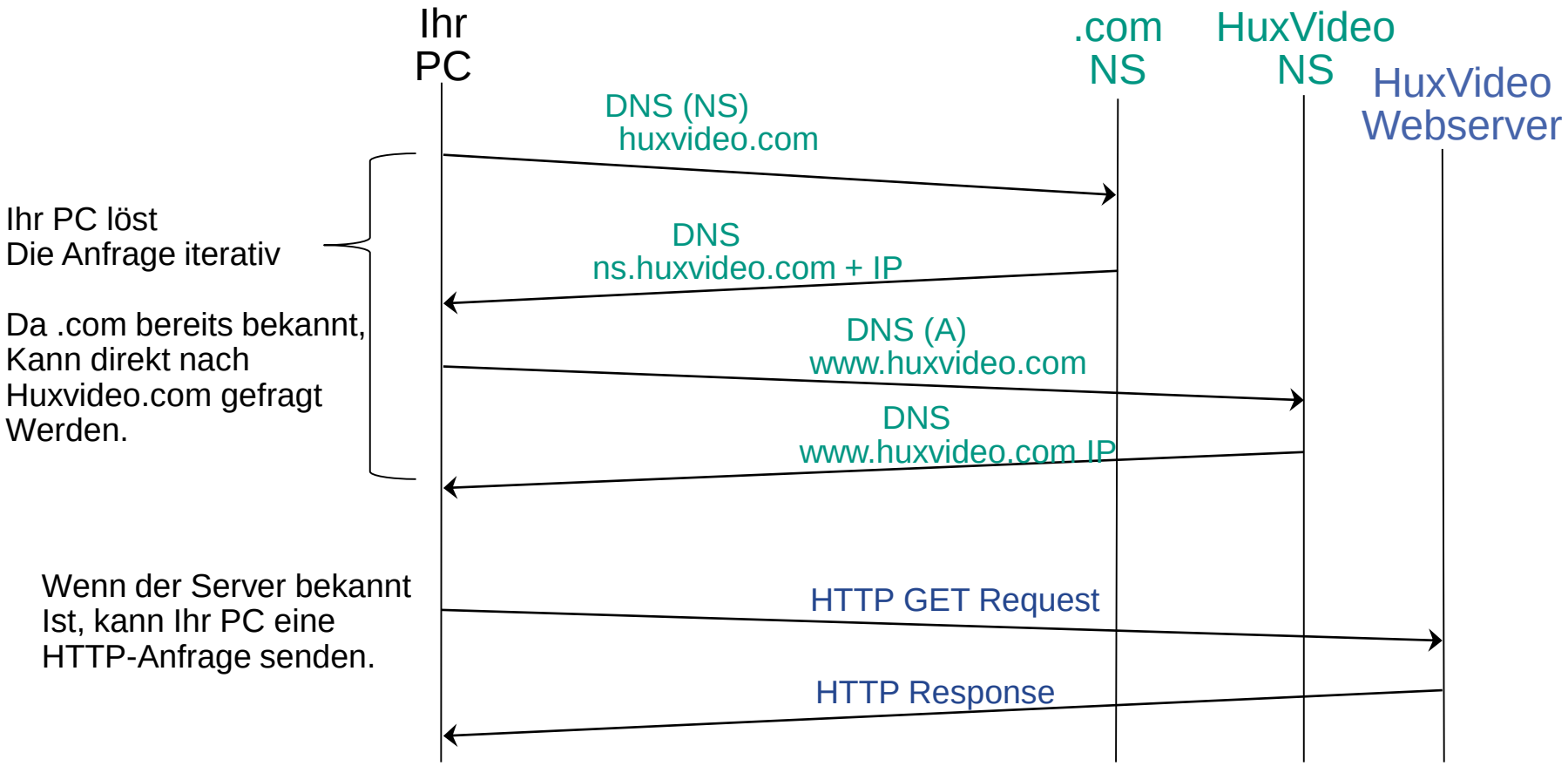
- Zu beachten:
  - DNS-Einträge werden gecacht – Es kann Stunden dauern, bis die Änderungen durch das Netz propagiert sind

(c) Aufruf der Startseite



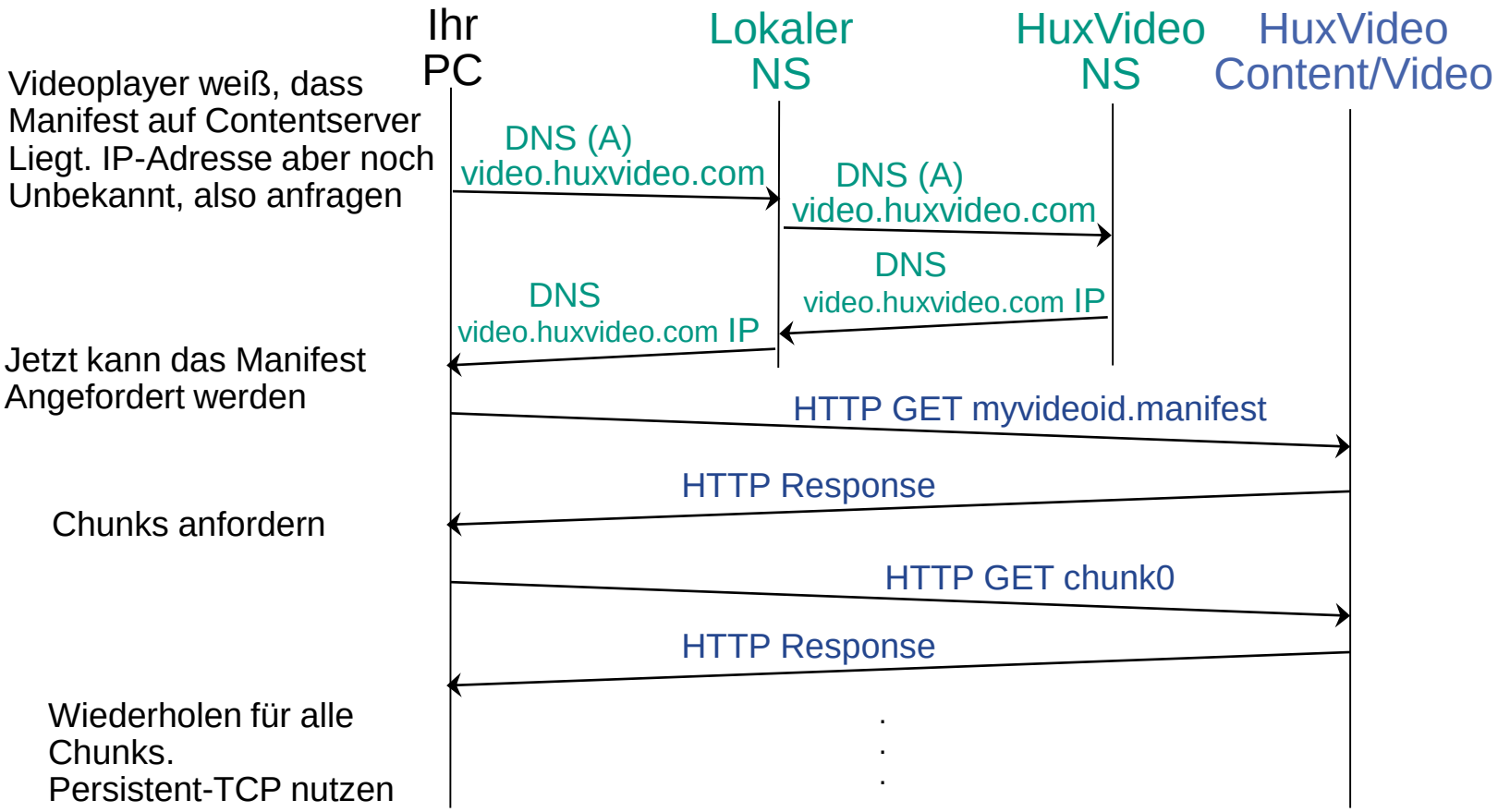
Hinweis:
Aufbau von TCP-Verbindungen ignoriert

(d) Alternative: Iterative DNS-Abfrage



Hinweis:
Aufbau von TCP-Verbindungen ignoriert

(e) DASH



Alternative:
Manifest liegt auf HV Webserver

Alternative:
Chunk-URLs auf selbem Server, aber anderer Domain
(in dem Fall zusätzliche Namensauflösung vor erstem Chunk-Request)

Hinweis:
Aufbau von TCP-Verbindungen ignoriert

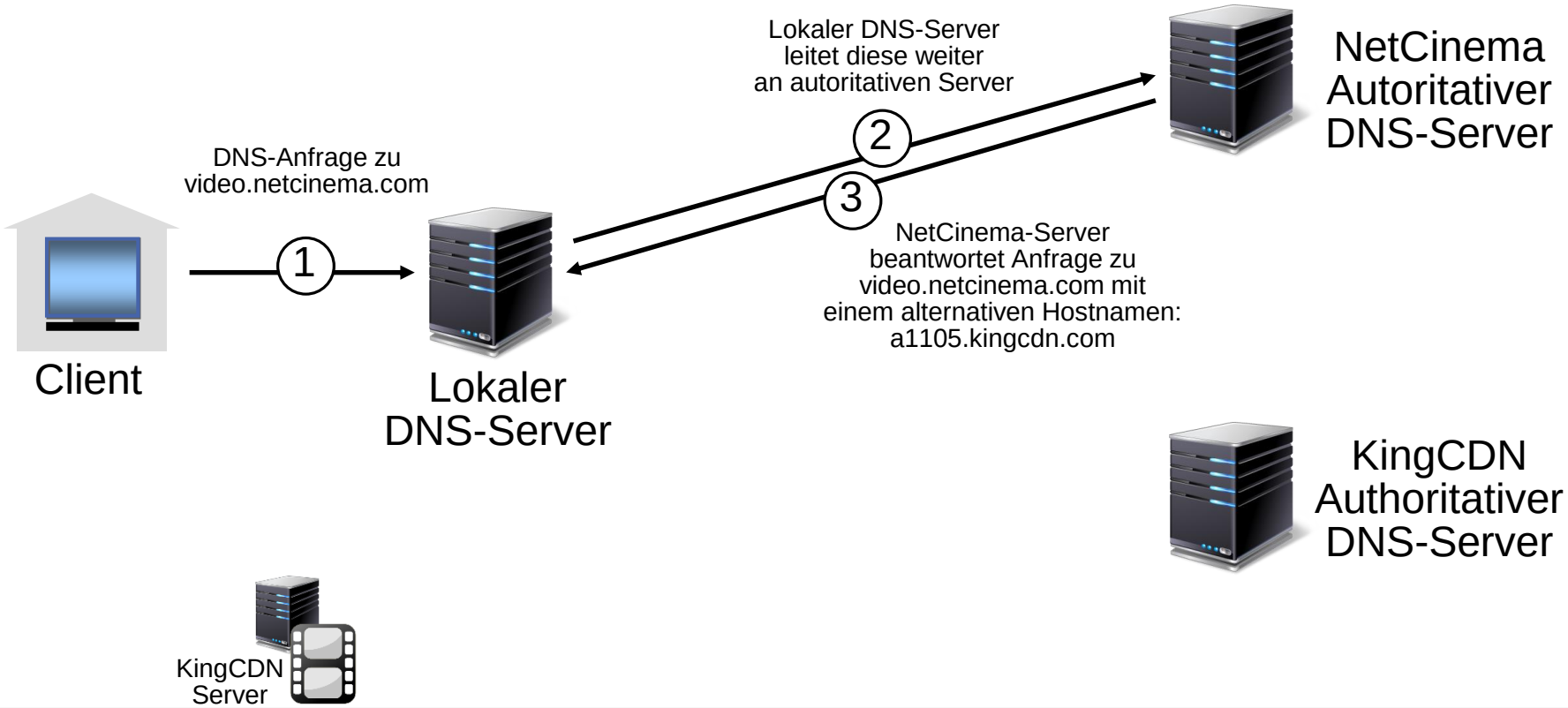
(f) Umzug auf KingCDN

- Kopieren der Video-Chunks auf die Server von KingCDN

Funktionsweise CDN

Vorlesung

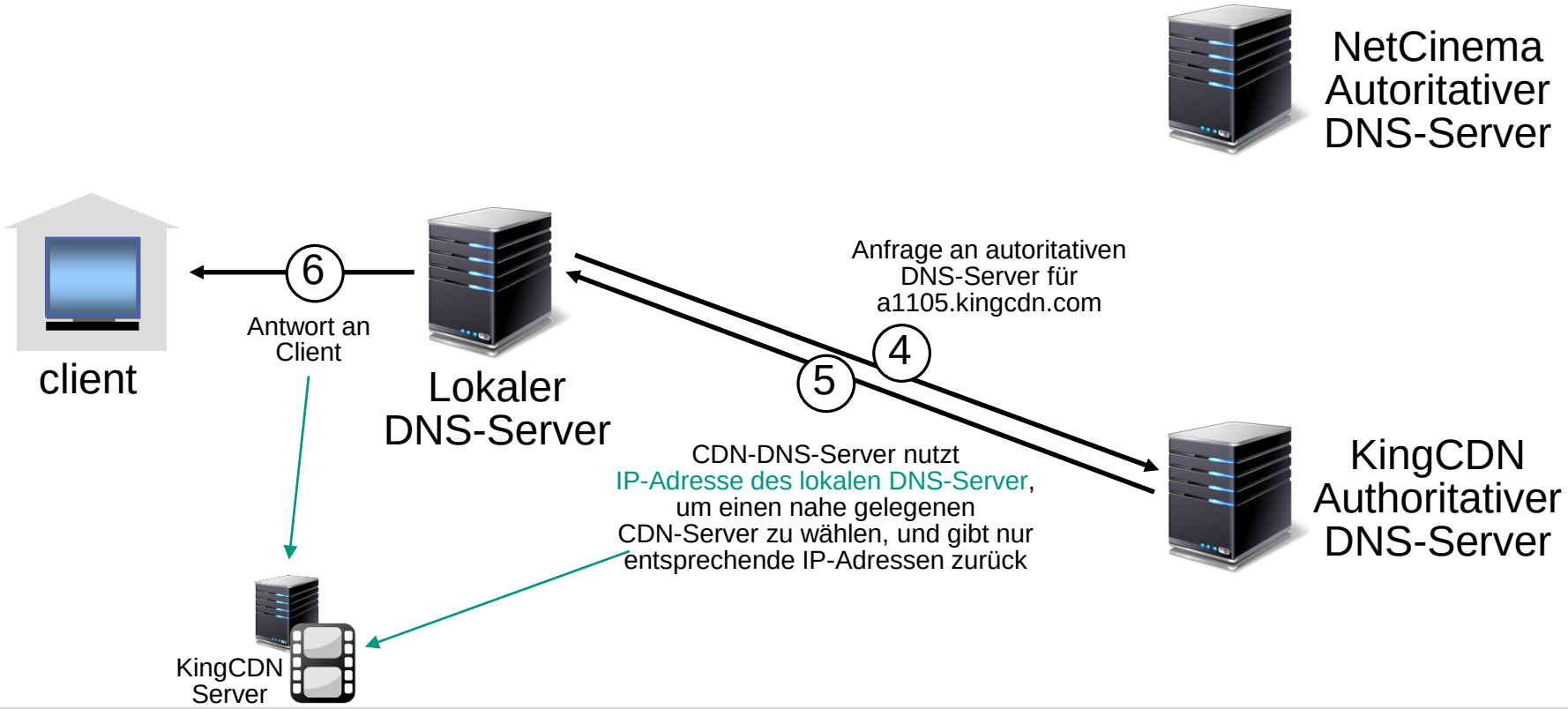
- Client kennt Adresse eines Chunks: http://video.netcinema.com/6Y7B2
  - video.netcinema.com muss noch aufgelöst werden
- Ziel: Umleiten der Anfrage auf lokalen CDN-Server



Funktionsweise CDN

Vorlesung

- Client kennt Adresse eines Chunks: http://video.netcinema.com/6Y7B2
  - video.netcinema.com muss noch aufgelöst werden
- Ziel: Umleiten der Anfrage auf lokalen CDN-Server



(f) Umzug auf KingCDN

- Kopieren der Video-Chunks auf die Server von KingCDN
- video.huxvideo.com wird umkonfiguriert (CNAME auf KingCDN-Domain)
  - DNS-Caching beachten!
- Alle URLs zu Manifests müssen auf content.huxvideo.com zeigen; alle URLs zu Chunks auf video.huxvideo.com
- Hinweis zu (g): Am Browsen der Website ändert sich nichts!

(g) DASH über CDN

Videoplayer weiß, dass Manifest auf Contentserver Liegt. IP-Adresse aber noch Unbekannt, also anfragen

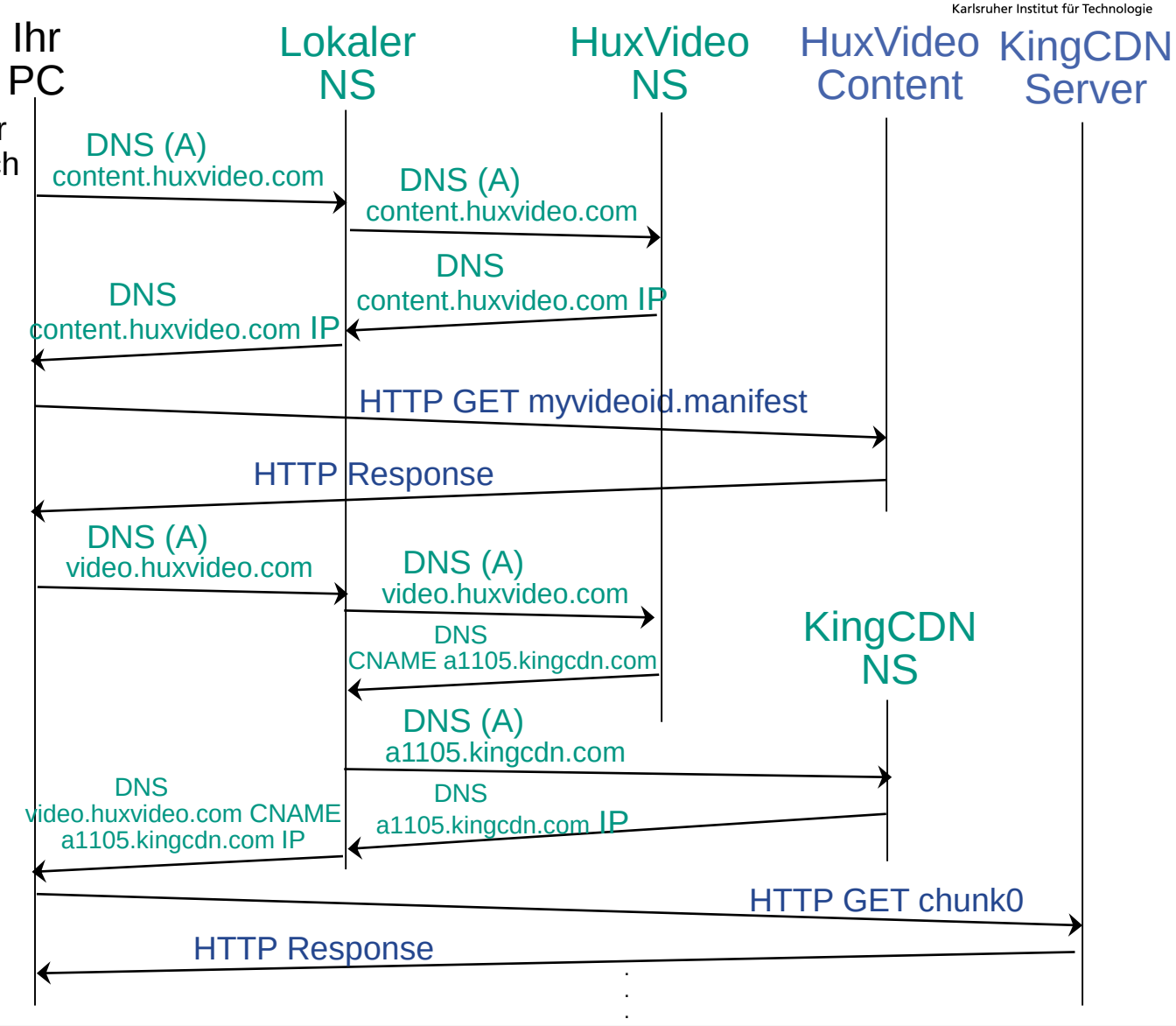
Jetzt kann das Manifest Angefordert werden

Manifest zeigt auf andere Domain -> auflösen

Lokaler NS kennt Kingcdn.com NS
(wenn nicht: an .com NS fragen)

Jetzt können Chunks Angefordert werden

Hinweis:
Aufbau von TCP-Verbindungen ignoriert



Einführung in Rechnernetze – Übungsblatt 2

1. HTTP-Quiz
2. DNS in der Praxis
3. CDN, DNS, und DASH
4. SMTP in der Praxis
5. Unterschiede im Nachrichtenaustausch
6. Telnet und NetCat
7. Socket-Tutorial

(a) Fully-Qualified Domain Name ermitteln

Zwei Möglichkeiten:

- Linuxkommandozeile: hostname -f
  - Schlägt fehl, wenn Domäne nicht konfiguriert
- Manuell
  - Öffentliche IP-Adresse ermitteln
    - Achtung: Im Heimnetz üblicherweise die WAN-IP-Adresse des Routers!
    - z.B. Services wie https://whatismyipaddress.com/
  - Reverse-Lookup durchführen
    - dig -x 129.13.40.10

(b) Mail-Server des Empfängers

- Empfänger sei: dein.name@student.kit.edu

- dig student.kit.edu MX
  - Name des Mail-Servers: scc-mailin-cs-01.scc.kit.edu

(c) Versenden der Mail

```
$ telnet scc-mailin-cs-01.scc.kit.edu smtp

Trying 2a00:1398:9:f710::810d:ae8c...
Connected to scc-mailin-cs-01.scc.kit.edu.
Escape character is '^]'.
220 scc-mailin-cs-01.scc.kit.edu ESMTP Thu, 11 May 2017 11:10:43 +0200
HELO myhost.tm.kit.edu
250 scc-mailin-cs-01.scc.kit.edu Hello myhost.tm.kit.edu [2a00:1398:[redacted]]
MAIL FROM: Tim Gerhard <tim.gerhard@kit.edu>
250 OK
RCPT TO: Tim Gerhard <tim.gerhard@kit.edu>
250 Accepted
DATA
354 Enter message, ending with "." on a line by itself
Hey, wie gehts?
.
250 OK id=1d8k7s-0002HO-2f
QUIT
221 scc-mailin-cs-01.scc.kit.edu closing connection
Connection closed by foreign host.
```

(d) E-Mail mit Zusatzoptionen

- Das ist Teil des E-Mail-Formats, nicht von SMTP
  - Eingabe von DATA muss um einen Header erweitert warden

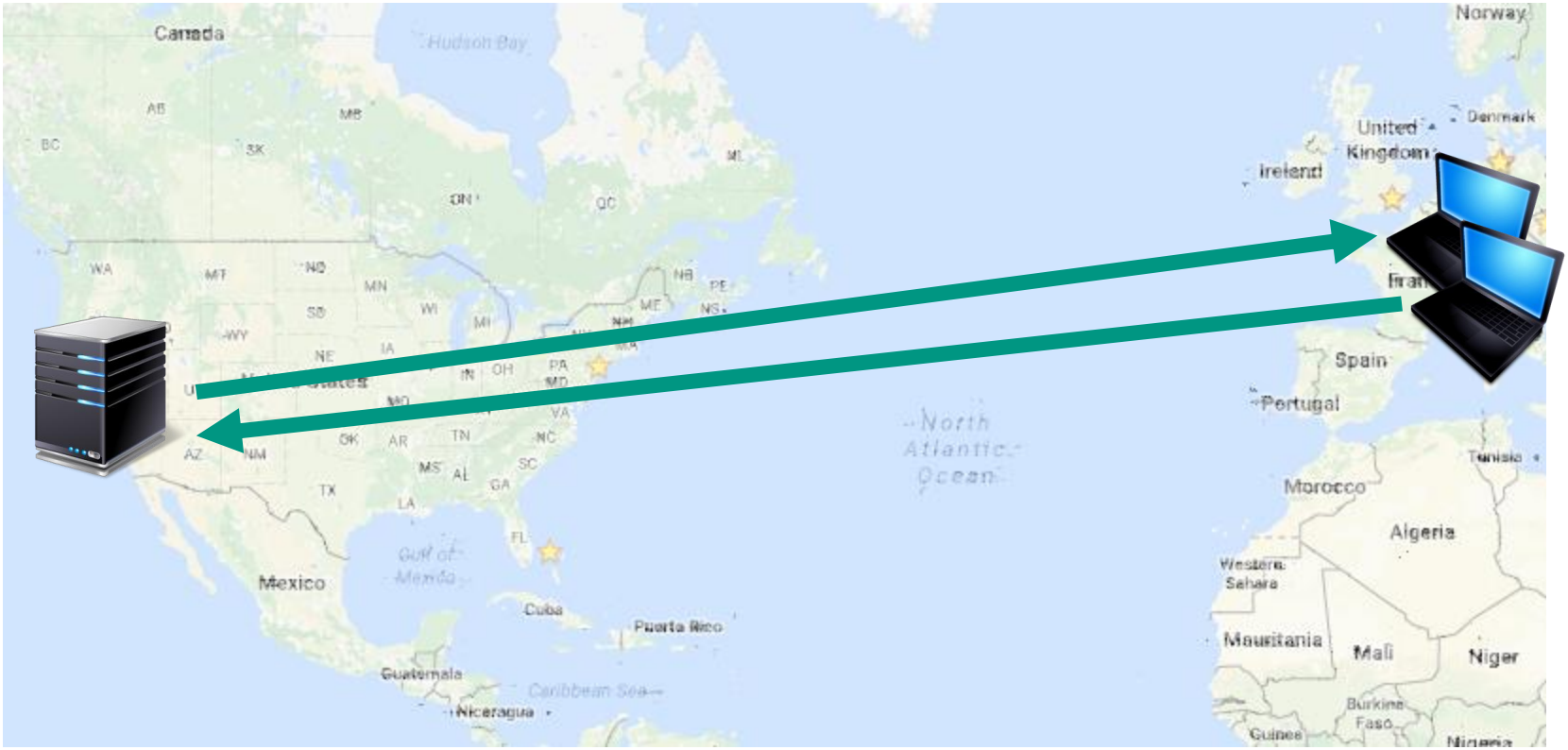
```
DATA
354 Enter message, ending with "." on a line by itself
Subject: Guten Morgen
To: Tim Gerhard <tim.gerhard@kit.edu>
From: Tim Gerhard <tim.gerhard@kit.edu>
CC: Matthias Flittner <flittner@kit.edu>

Hey, wie gehts?
.
250 OK id=1d8k7s-0002HO-2f
```

Einführung in Rechnernetze – Übungsblatt 2

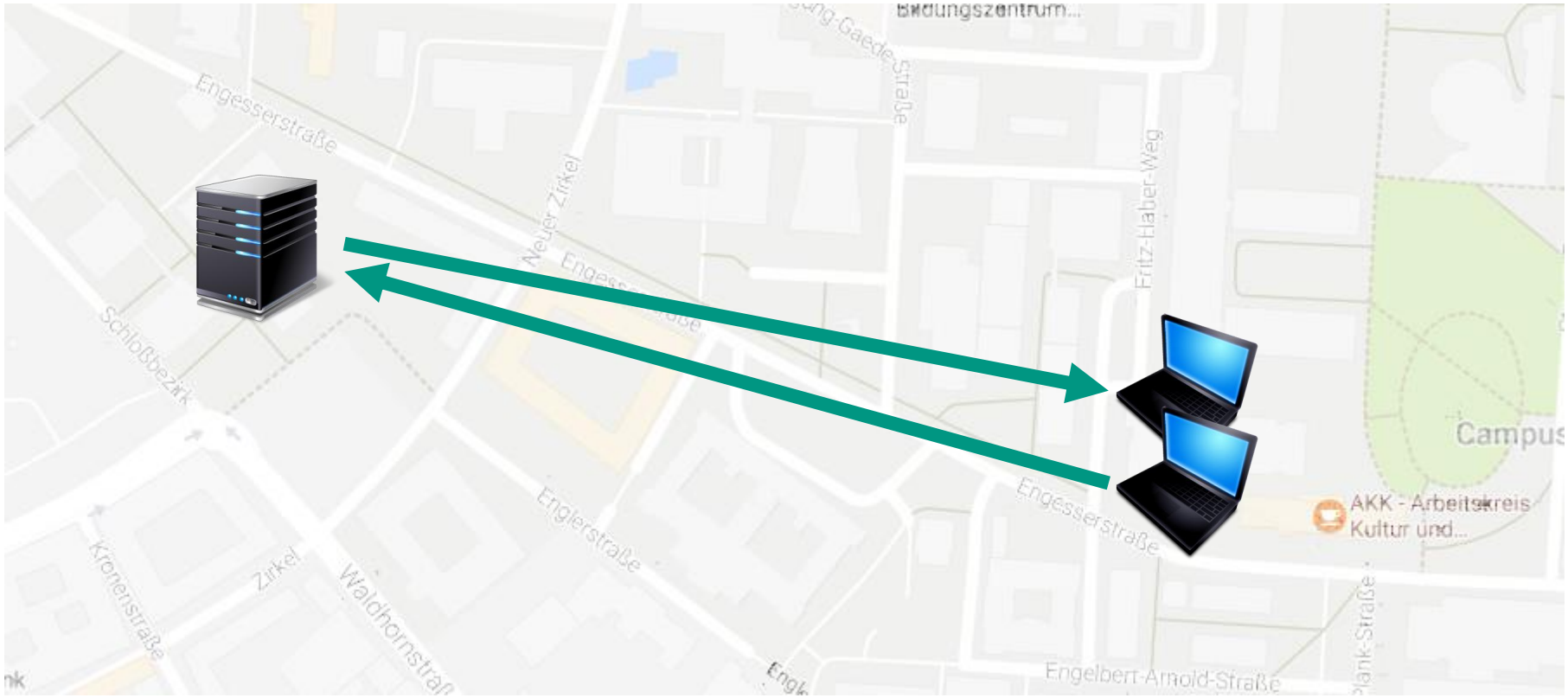
1. HTTP-Quiz
2. DNS in der Praxis
3. CDN, DNS, und DASH
4. SMTP in der Praxis
5. Unterschiede im Nachrichtenaustausch
6. Telnet und NetCat
7. Socket-Tutorial

(a) WhatsApp

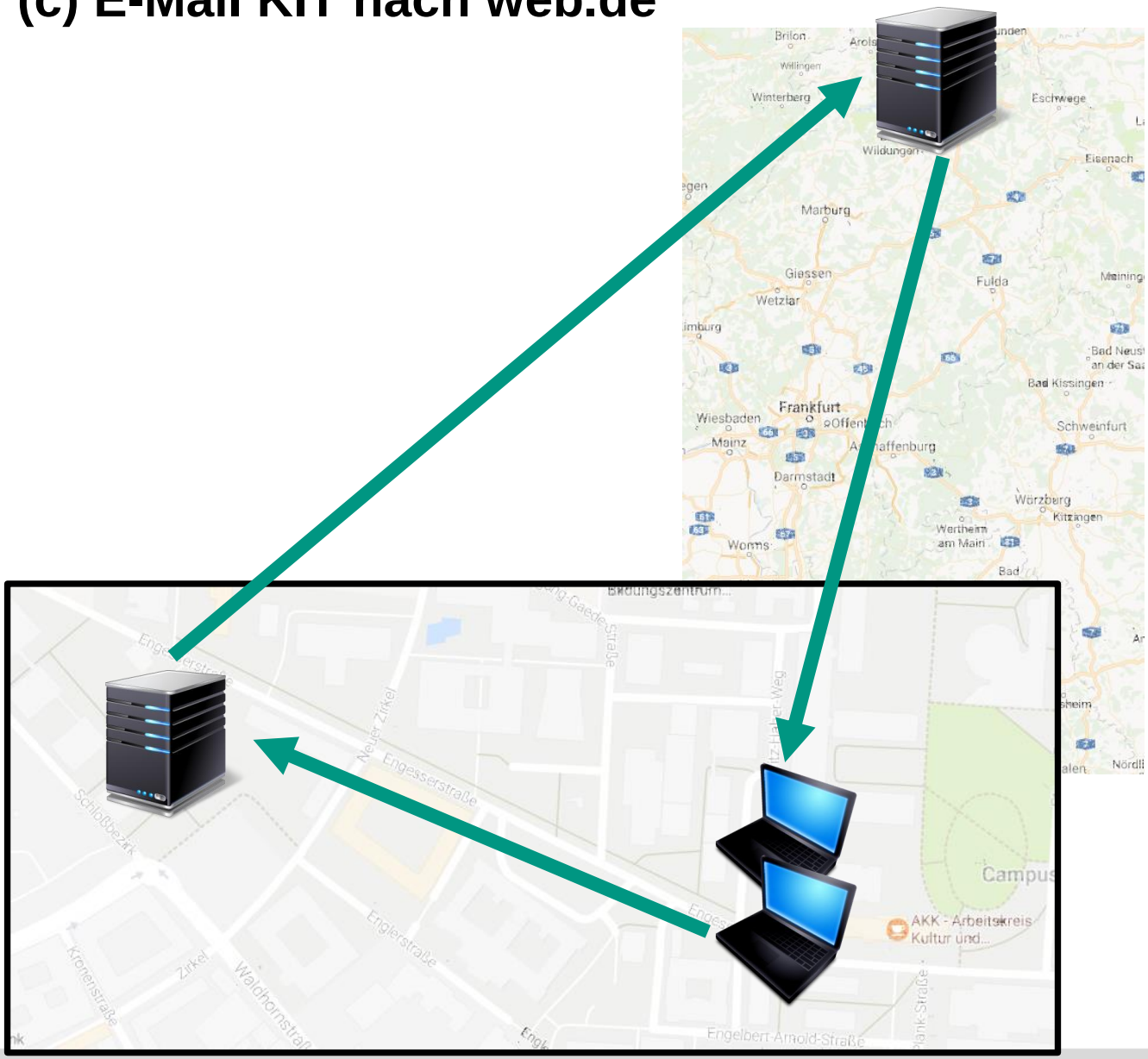
(a) WhatsApp


Canada
Hudson Bay
Norway
Denmark
United Kingdom
Ireland
Spain
Portugal
North Atlantic Ocean
Morocco
Algeria
Western Sahara
Mauritania
Mali
Niger
Burkina Faso
Guinea
Tunisia
Mexico
Gulf of Mexico
Cuba
Puerta Rico
Guatemala
Nicaragua
Caribbean Sea

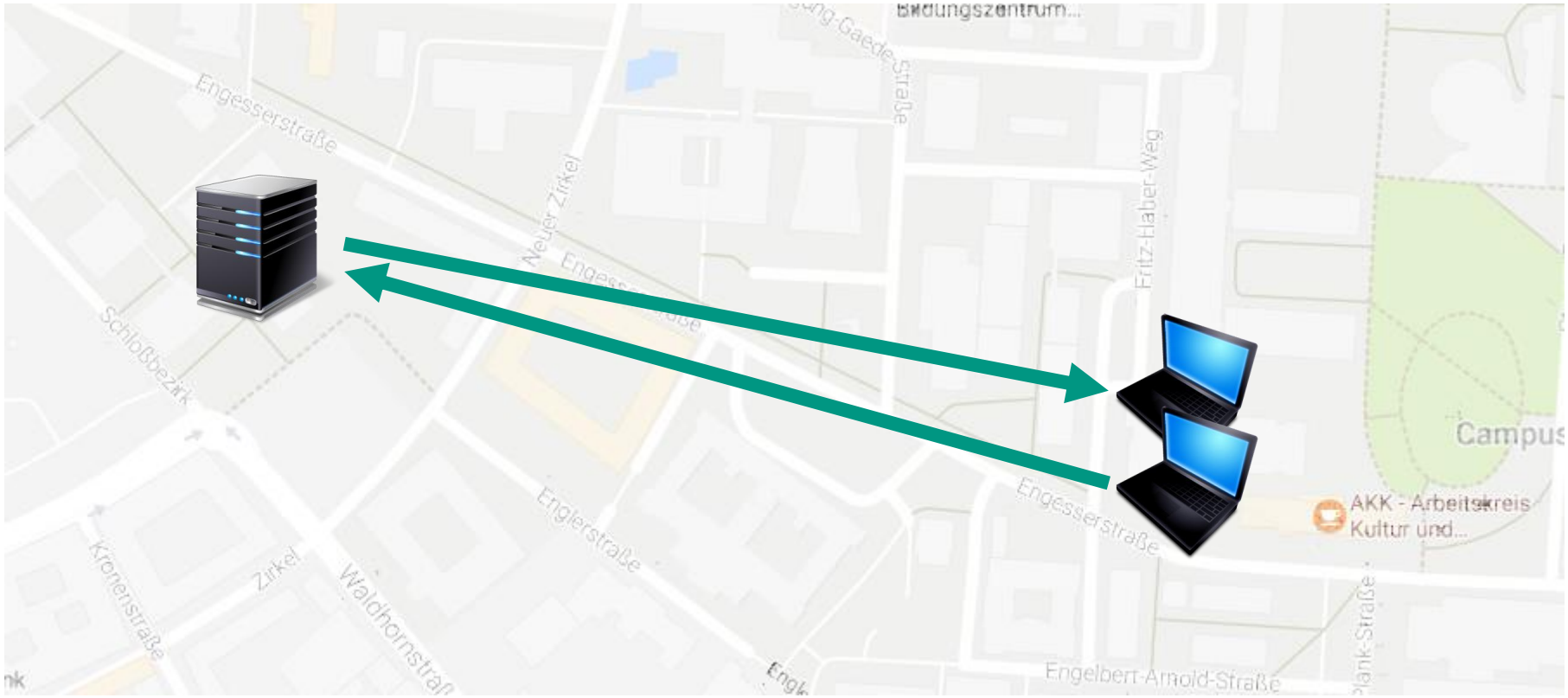
(b) E-Mail KIT-Intern



(c) E-Mail KIT nach web.de



(d) XMPP KIT-intern



Übertragungsfehler (Netzwerk)

- Paketverlust
- Paket Reordering
- Bitfehler

Nächste Vorlesung

Übertragungsfehler (Endsysteme)

- Absturz eines Clients
  - Sender vor Senden: automatisch senden nach Neustart, oder Neueingabe durch Nutzer erforderlich
  - Sender nach Senden: egal
  - Empfänger vor Empfang: Server wartet, bis Empfänger wieder online
  - Empfänger nach Empfang: Nachricht ist verloren oder gespeichert

- Absturz eines Servers
  - Vor Empfang: Sender muss warten
  - Nach Empfang, vor Senden: Nachricht ist verloren oder zwischengespeichert.
  - Nach Senden: egal

Einführung in Rechnernetze – Übungsblatt 2

1. HTTP-Quiz
2. DNS in der Praxis
3. CDN, DNS, und DASH
4. SMTP in der Praxis
5. Unterschiede im Nachrichtenaustausch
6. Telnet und NetCat
7. Socket-Tutorial

Telnet und NetCat

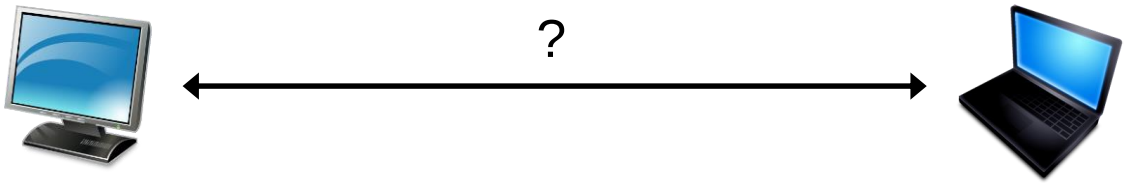
- Telnet ist Protokoll mit deutlich mehr Features
  - Remote-Login, Dateiübertragung, …
  - Unsicher!
  - Raw-TCP-Funktionalität (wie in den Übungen verwendet) ist ein Fallback-Modus

- NetCat
  - Tool zum Bedienen von vielen Transportprotokoll-Features
  - Raw-TCP-Funktionalität vorgesehen

Herausforderung: Heterogenität

- Probleme in heterogenen Netzen können entstehen durch
  - unterschiedliche Programmiersprachen
  - unterschiedliche Typsysteme und Aufrufkonventionen
  - unterschiedliche Repräsentation der Daten

- Herausforderung: Einigung auf gemeinsame Transfersyntax (TODO: check)



- Relevant bei allen Arten von Datentransport!

NetCat und HTTP

- Problem mit Webservern
  - NetCat gibt Standardeingabe direkt an TCP-Socket weiter
    - LF als LF
  - Telnet ändert ggf. Codierung
    - LF zu CR-LF

- HTTP erwartet per Definition CR-LF
  - LF ist undefiniertes Verhalten
    - Kann zu Protokollfehlern führen (z.B. bei www.kit.edu)
    - Muss aber nicht (z.B. bei i72tmdjango.tm.kit.edu)

Einführung in Rechnernetze – Übungsblatt 2

1. HTTP-Quiz
2. DNS in der Praxis
3. CDN, DNS, und DASH
4. SMTP in der Praxis
5. Unterschiede im Nachrichtenaustausch
6. Telnet und NetCat
7. Socket-Tutorial

Multiplexen / Demultiplexen

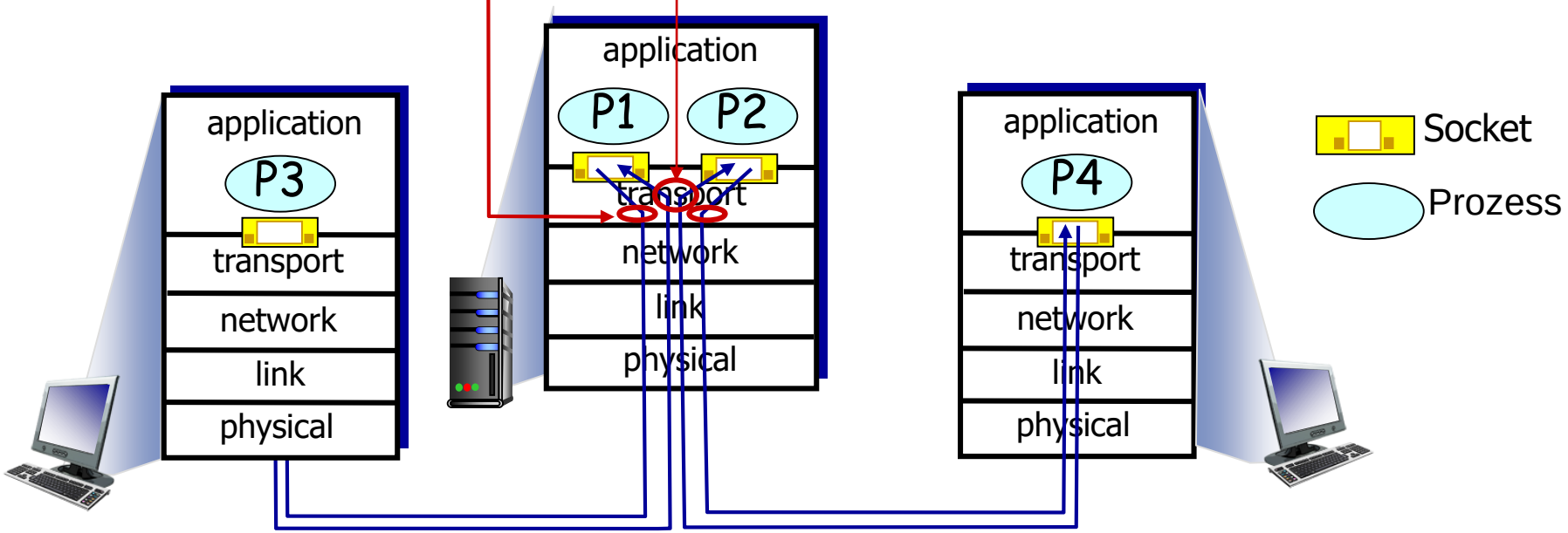
Vorlesung

Multiplexing beim Sender:

Daten von mehreren Sockets, Transportkopf hinzufügen

Demultiplexing beim Empfang

Information im Transportkopf nutzen, um empfangene Segmente an die korrekten Sockets auszuliefern



Verbindungsloses Demultiplexen

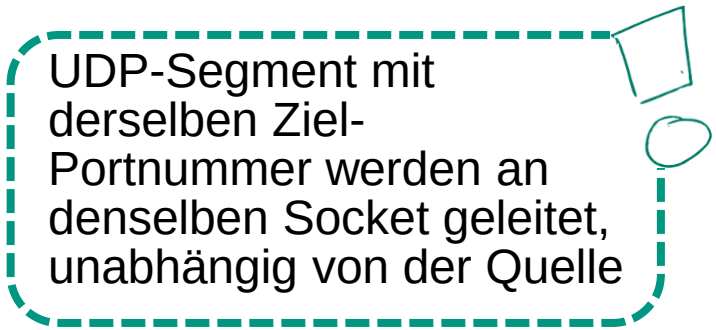
- Sockets verfügen über Host-lokale Portnummer

```
DatagramSocket mySocket1 = new DatagramSocket(12534);
```

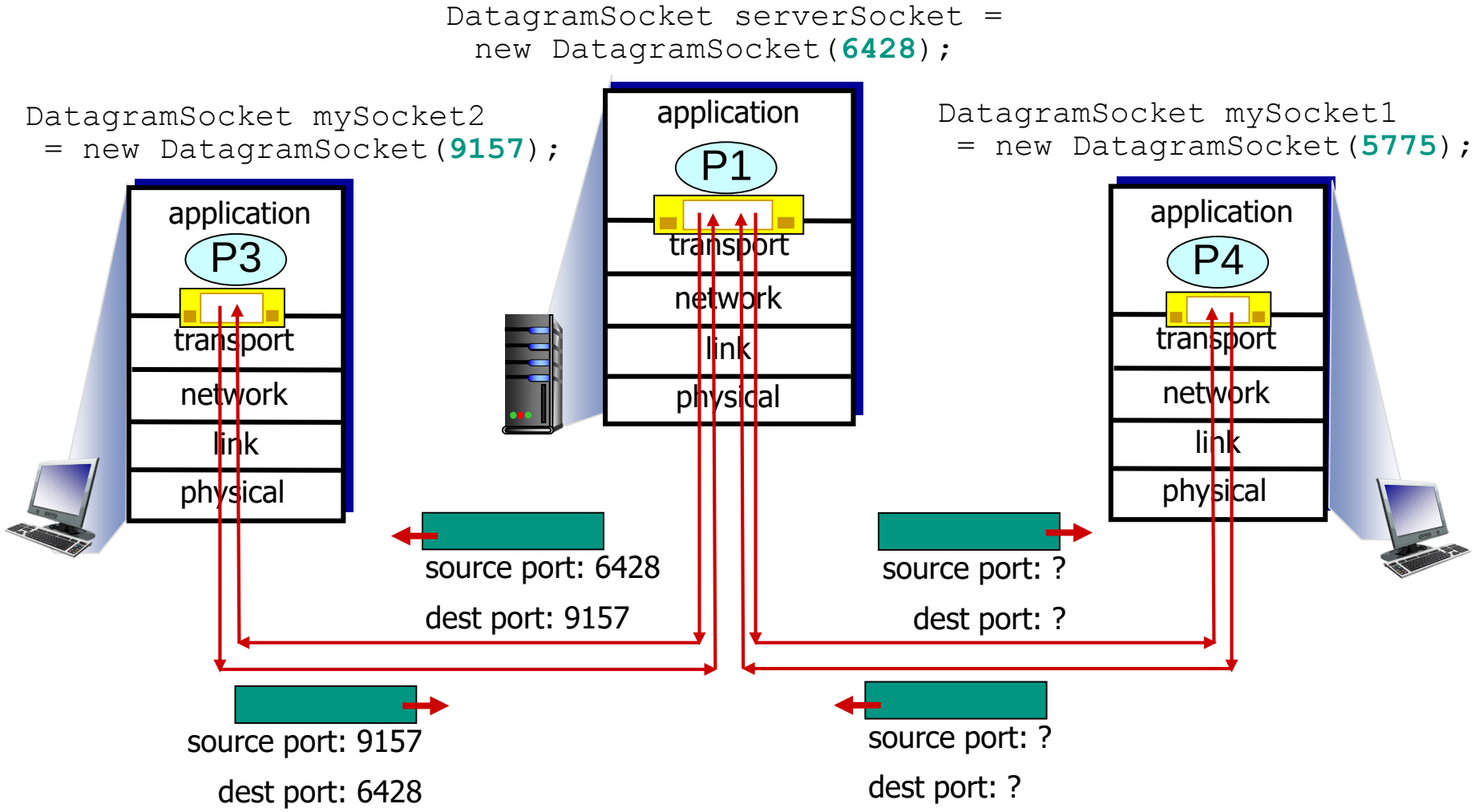
- Beim Erstellen von UDP-Segmenten müssen Ziel-IP-Adresse und Portnummer angegeben werden

```
Datagram mySegment1 = new Datagram(myDestIp, 12534);
```

- Host empfängt UDP-Segment
  - Prüfe Ziel-Portnummer des Segments
  - Leite Segment an Socket mit dieser Portnummer weiter



Verbindungsloses Demultiplexen: Beispiel



Verbindungsorientiertes Demultiplexen

- TCP-Sockets werden durch ein 4-Tupel identifiziert
  - Quell-IP-Adresse
  - Quell-Portnummer
  - Ziel-IP-Adresse
  - Ziel-Portnummer

- Der Zielhost verwendet alle vier Werte um ein empfangenes Segment an den entsprechenden Socket weiterzuleiten

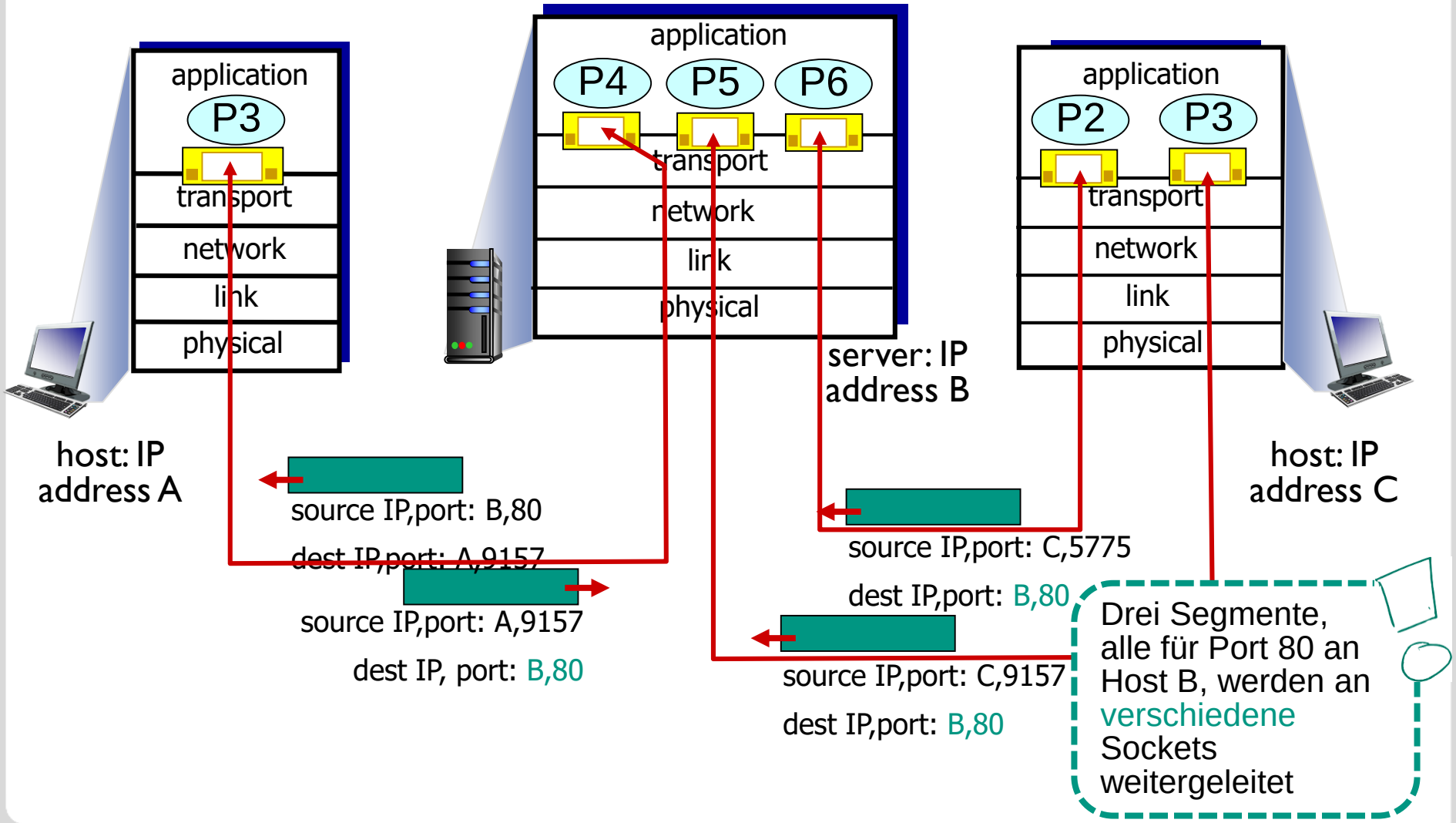
- Ein Server kann viele simultane TCP-Sockets verwenden
  - Jeder Socket wird durch sein eigenes 4-Tupel identifiziert

Die Identifikation beinhaltet die eigene Adresse. Außerdem ist ein Socket an sein Netzwerk-Interface gebunden. Dies ist ein Problem für mobile Endgeräte!

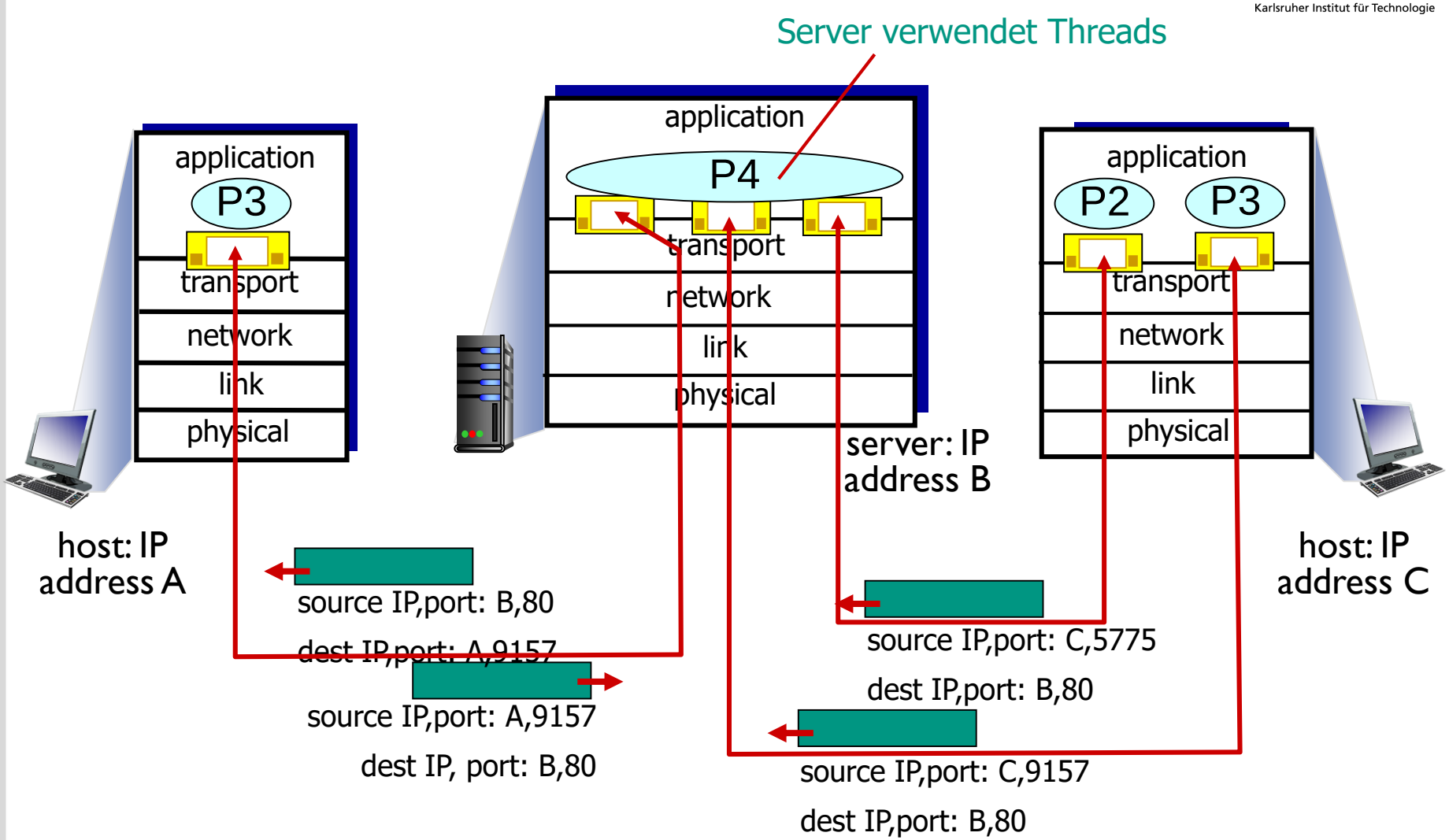
Webserver nutzen einen eigenen Socket für jeden verbundenen Client.

Non-persistent HTTP: Eigener Socket für jede Anfrage – Speicheraulastung!

Verbindungsorientiertes Demultiplexen: Beispiel



Verbindungsorientiertes Demultiplexen: Beispiel

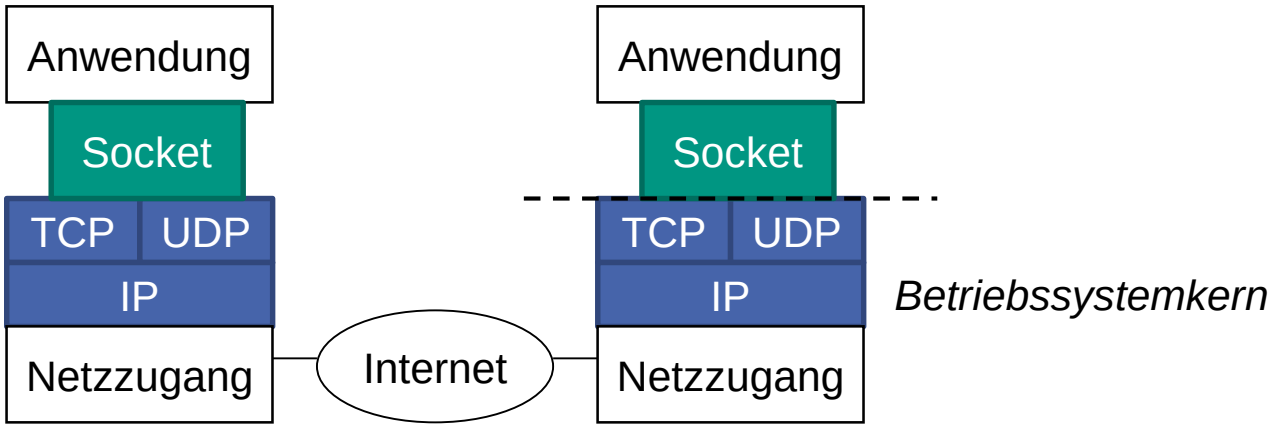


Vergabe der Portnummern

- Well known port numbers (0 – 1023)
  - Vergabe durch IANA (Internet Assigned Numbers Authority)
  - Nutzung nur durch System- oder privilegierte Prozessen
- Registered port numbers (1024 – 49151)
  - Registrierung durch IANA
  - Nutzung durch Nutzer- oder gewöhnliche Prozesse
- Dynamic and/or private ports (49152 – 65535)
  - Keine Registrierung vorgesehen

Sockets: Programmierschnittstelle

- Sockets
  - Schnittstelle zwischen Anwendung und Betriebssystem
  - Eingeführt in Berkeley UNIX, 1981
  - Entwickelt für die Programmiersprache C
  - In der Praxis weit verbreitet
  - Flexible Schnittstelle für die Nutzung verschiedener Kommunikationsprotokolle
    - Interprozesskommunikation, TCP, UDP, IP, X.25 ...
  - Socket-Objekt
    - Kommunikationsendpunkt für bidirektionalen Datenaustausch

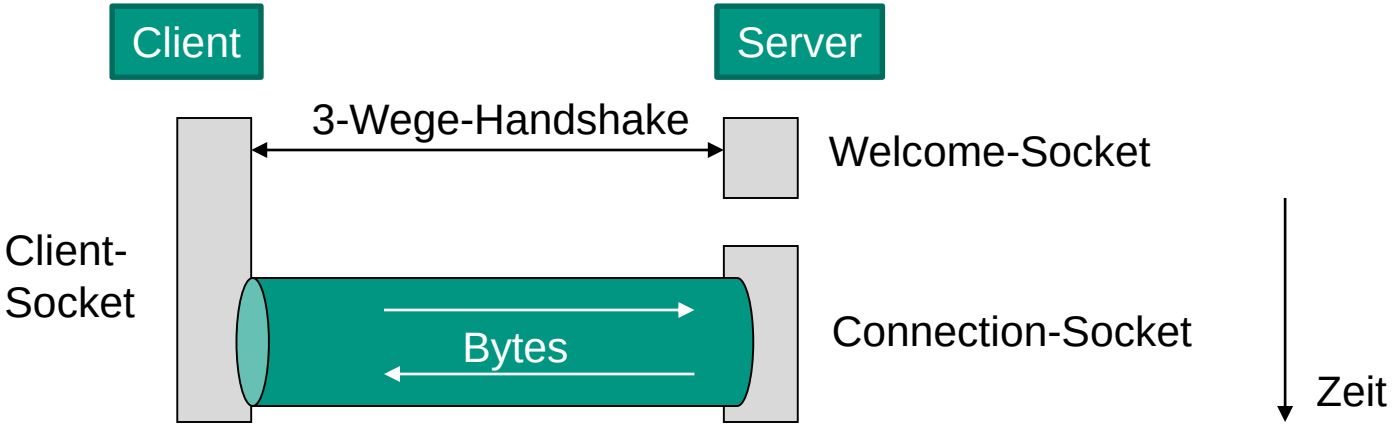


Sockets: Eigenschaften

- Sockets werden von Anwendungen erzeugt, genutzt und wieder freigegeben
- Adressierung auf Anwendungsebene über Ports
- Sockets ermöglichen das Lesen und Schreiben von Daten
  - Interne Adressierung erfolgt über Deskriptoren
    - Ablauf analog zu einfachem Dateizugriff (Filedeskriptor)
    - Transfer der Daten lokal oder über Netzwerk
- Sockets können empfangene oder zu sendende Daten kurzzeitig speichern
  - Sockets belegen Ressourcen

Socketprogrammierung mit TCP

- Server benutzt zwei Sockets
  - Welcome-Socket, über den er angesprochen wird
    - Muss ständig aktiv sein
  - Connection-Socket, der dann für den eigentlichen Datentransfer etabliert wird
- Clientprozess kann dann eine Verbindung zum Serverprozess initiieren
  - Kreiert ein Socket-Objekt, den Client-Socket
  - Parameter: Ziel-IP-Adresse, Ziel-Portnummer
  - 3-Wege-Handshake zum Aufbau der TCP-Verbindung transparent für Clientprozess



Wichtige Eigenschaften von TCP-Sockets

- Input/Output-Streams
  - TCP-Verbindungen bestehen aus zwei Streams (senden/empfangen)
  - Werden als Input/Output Streams an Prozesse gegeben (lesen/schreiben)
- Pufferung
  - Daten werden nicht immer sofort gesendet
    - Z.B. Nagle's Algorithmus
    - Vermeidet Overhead durch viele kleine Pakete
- Transparente Bedienung
  - Verbindungsaufbau nicht sichtbar
  - Keine Informationen, was Client bereits empfangen hat
    - Obwohl die Socket-Imlementierung das weiß
- Kernel-API ist standardisiert
- Programmiersprachen übersetzen diese in ihre eigene Konzepte
- Timeout
  - TCP-Verbindungen warden nach einem Timeout geschlossen

TCP-Sockets in Java (1/2)

```
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.BufferedReader;
import java.net.Socket;
```

- TCP-Sockets können jederzeit erzeugt werden:

```
String host = "i72tmdjango.tm.kit.edu";
int port = 8000;
Socket httpcli = new Socket(host, port);
```

- Bilden TCP-Stream auf InputStream und OutputStream ab
  - Diese müssen als Java-IO verarbeitet warden, z.B.:

```
PrintWriter out = new PrintWriter(httpcli.getOutputStream(), true);

BufferedReader in = new BufferedReader(
  new InputStreamReader(httpcli.getInputStream())
);
```

TCP-Sockets in Java (1/2)

- Senden von Daten:

```
out.println("GET /first.txt HTTP/1.1\r\nHost: i72tmdjango.tm.kit.edu\r\n\r\n");
```

- Empfangen von Daten:

```
String ln;
while ((ln = in.readLine()) != null) {
  System.out.println(ln);
}
```

Diese Methoden können Fehler werfen; diese müssen mit try/catch gefangen werden.

EXAMPLE

```
ghd@rchnr:~/cl$ java Reader
HTTP/1.0 200 OK
Date: Tue, 16 May 2017 07:26:32 GMT
Server: WSGIServer/0.1 Python/2.7.9
X-Frame-Options: SAMEORIGIN
Content-Type: text/html; charset=utf-8

Herzlichen Glueckwunsch!
```

Eine stabile Anwendung muss viele Fehlerfälle abdecken. Je nach Art des Protokolls muss auf die Eigenschaften des Sockets Rücksicht genommen werden.

TCP-Server-Sockets in Java

- Server-Socket wartet auf eingehende Verbindungen:

```
import java.net.Socket;
import java.net.ServerSocket;
```

```
int port = 8000;
ServerSocket serverSocket = new ServerSocket(port);
```

- Mit accept() kann auf eine neue Verbindung gewartet werden.
  - Die neue Verbindung ist ein Socket, wie der des Clients:

```
Socket clientSocket = serverSocket.accept();
```

ServerSockets und Sockets sollten, wenn nicht mehr benötigt, mittels close() geschlossen werden

Heute Abend: Kleiner Wettbewerb

- Die Aufgabe
  - Besiegen Sie *Nemesis* durch Nutzung von Sockets
  - Lösen von 50 Aufgaben in zwei Minuten

Sie

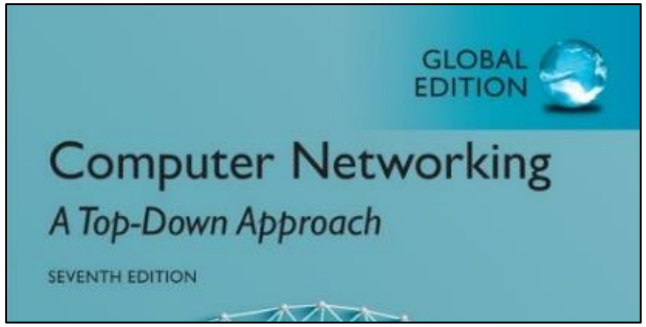
Wörter: `pvumr tilsxt ptxsli t prmuvpj`

Palindrom: `tilsxtprmuvpjpvumrptxslit`

Nemesis

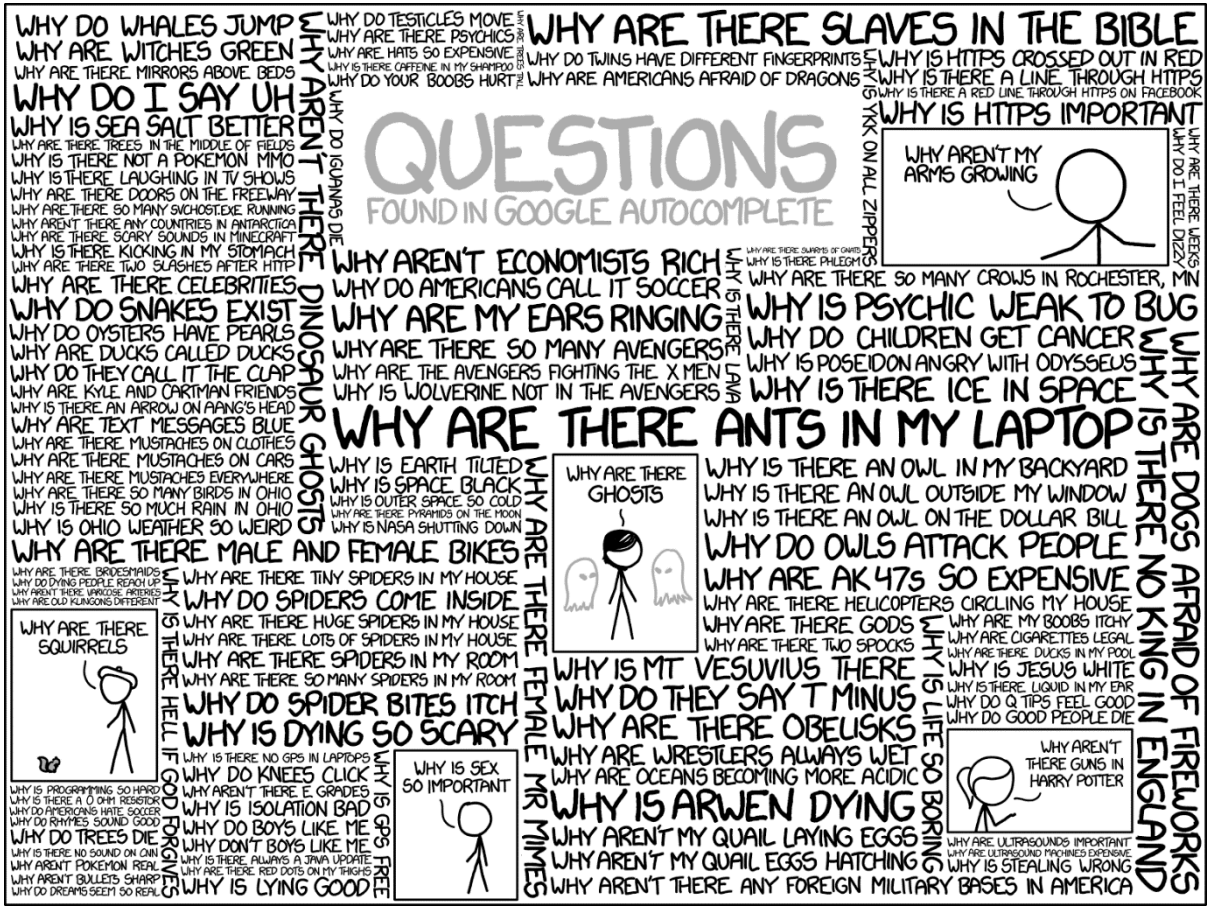
Fakten

- Mehr Infos auf dem 2. Übungsblatt
- Wettkampf läuft ab 17.05.17; 19:30 Uhr
  - Mehr Infos ab dann über ILIAS
- Siegerehrung in 3. Übung; 31.05.2017
- Benachrichtigung per @student.kit.edu



Hauptgewinn
!! Lehrbuch zur Vorlesung !!

Vielen Dank! Fragen?



- Nächste Rechnernetze-Übung am

31.05.2017