Work in progress

Rechnernetze (2)

Ute Bormann, TI2

2023-10-13

Inhalt

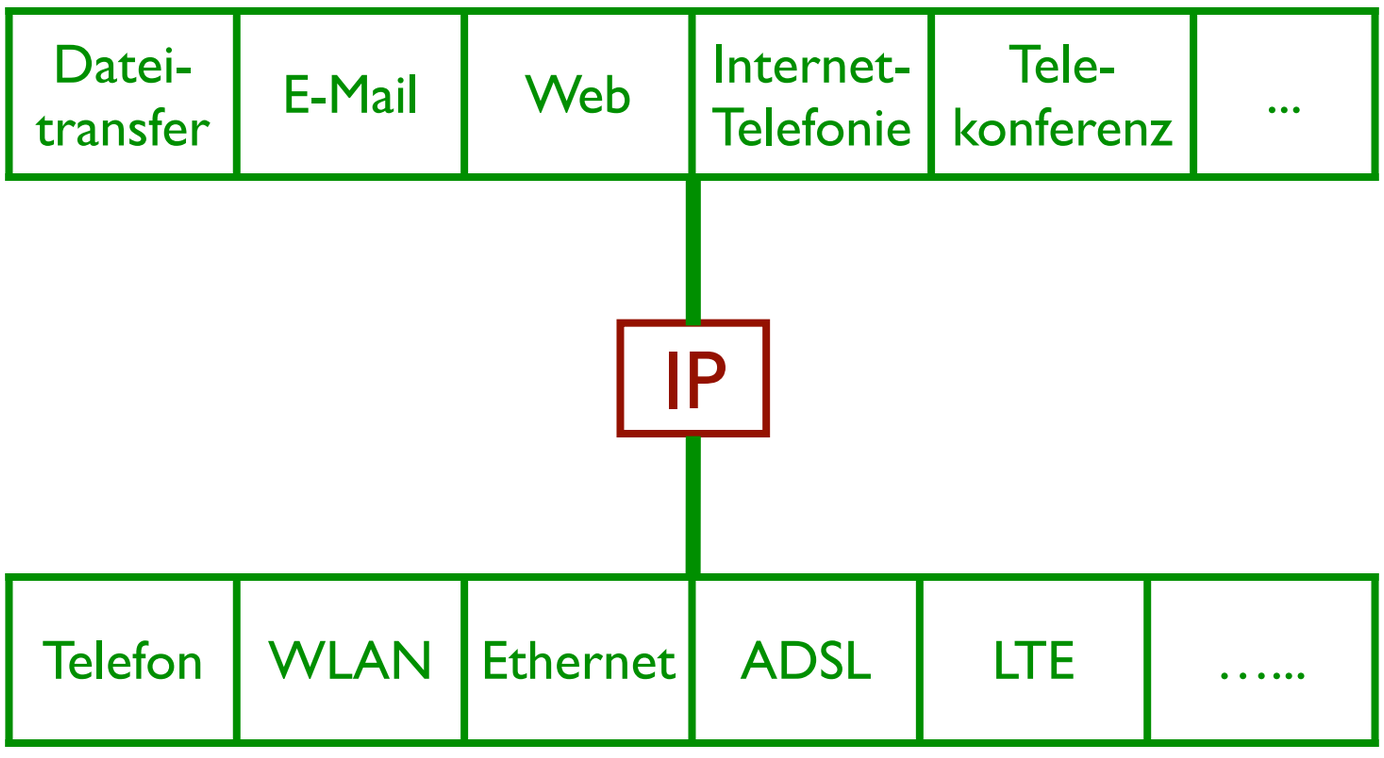
1. Kommunikationsarchitektur
2. Internet Protocol (IP)
3. Transmission Control Protocol (TCP) und Domain Name Service (DNS)
4. Web-Anwendung

Teil 1:
Kommunikationsarchitektur

24 Rechnernetze (2), Ute Bormann

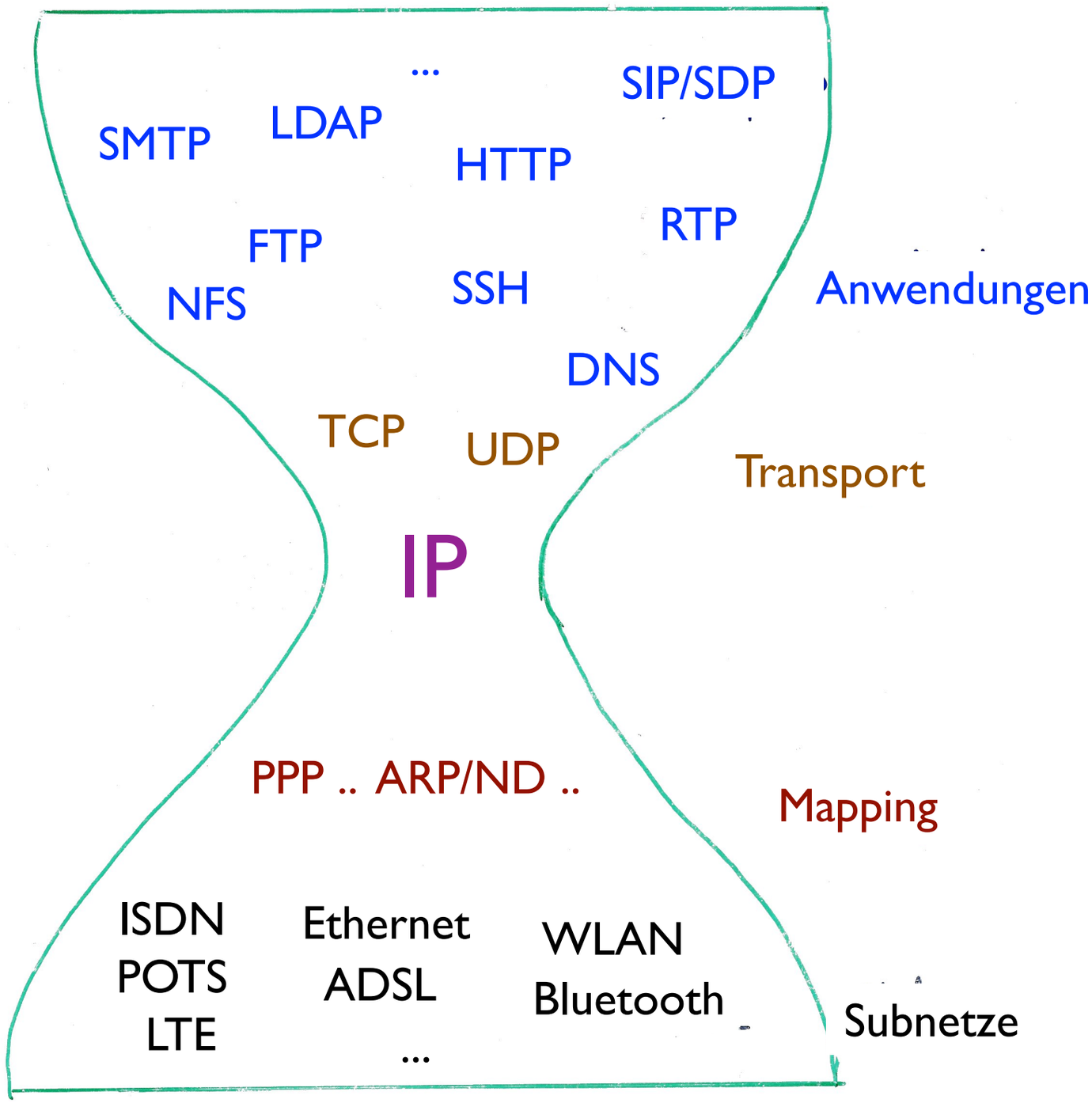
Die Bedeutung von IP

- Integration von Anwendungen
- Integration von Medien

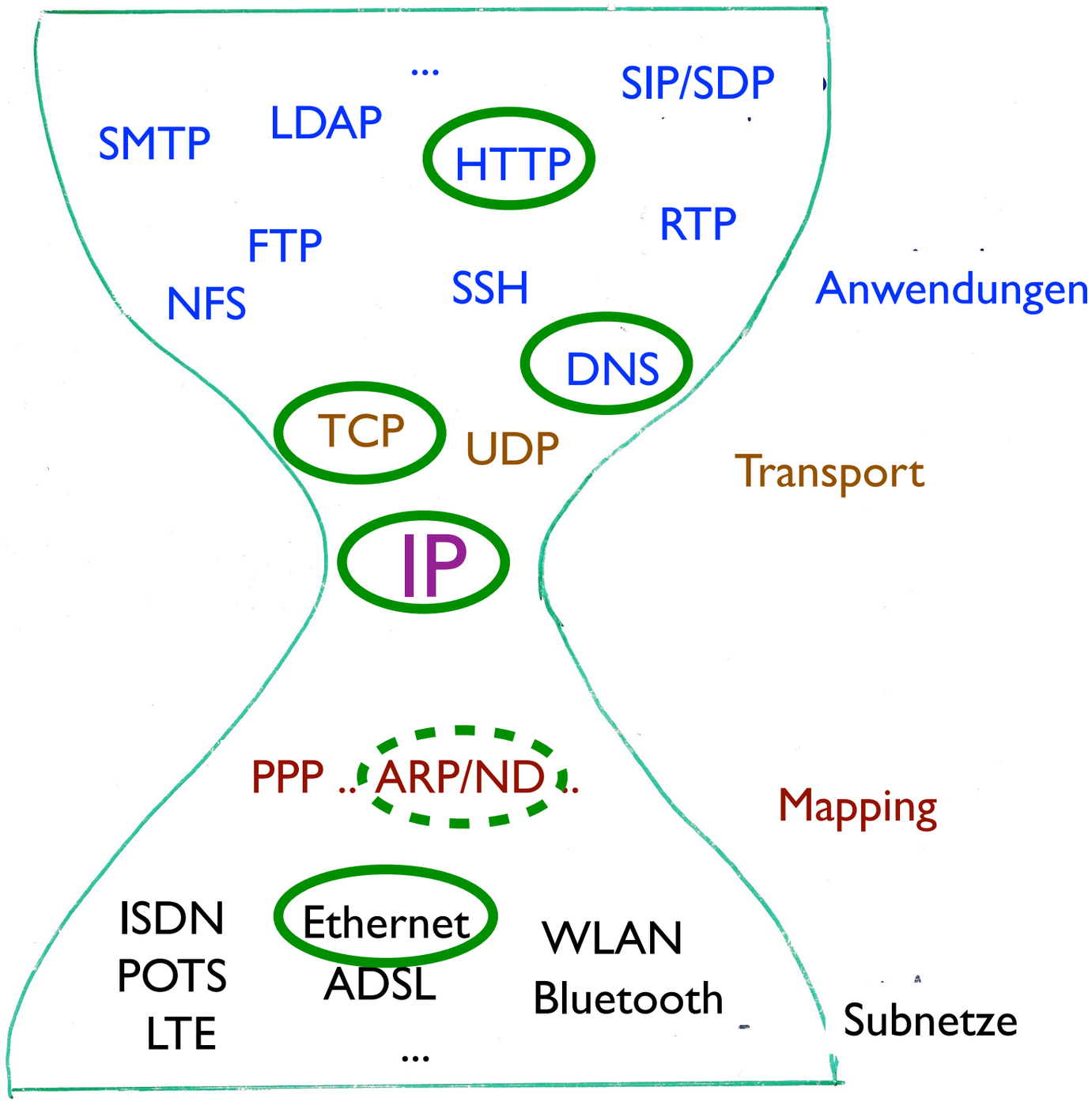


- Integration von Netztechnologien
  - netzübergreifende Adressierung
  - Paket-orientiertes Multiplexing

Die Bedeutung von IP (genauer)

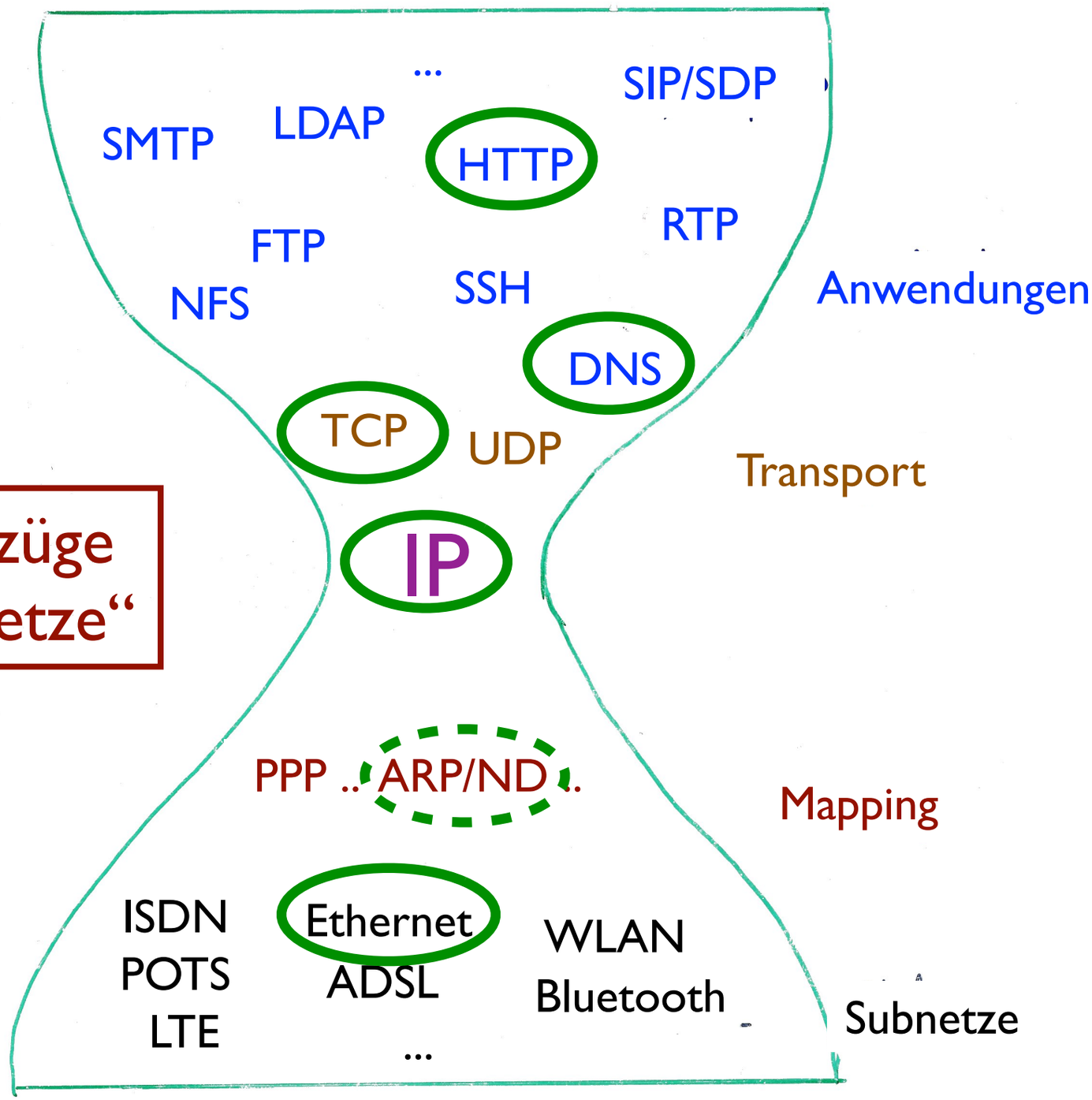


Die Bedeutung von IP (genauer)



⇒ Beispielprotokolle in Auszügen anschauen

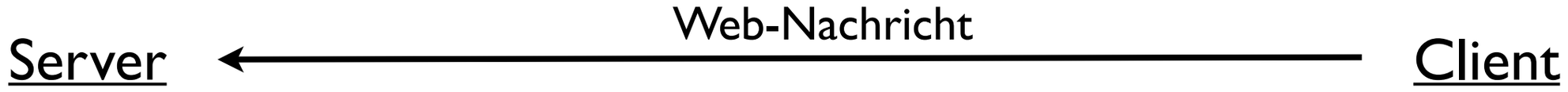
Die Bedeutung von IP (genauer)



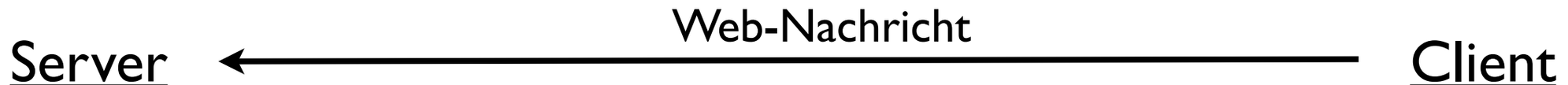
Größtenteils Auszüge aus LV „Rechnernetze“

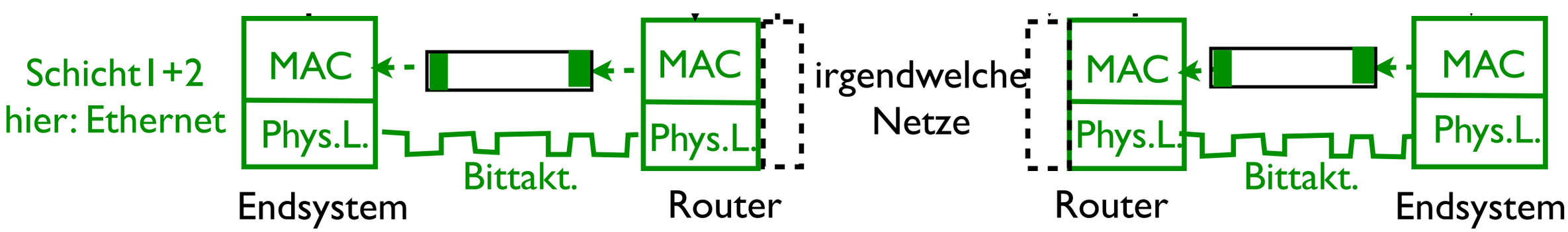
⇒ Beispielprotokolle in Auszügen anschauen

Vereinfachter Kommunikationsablauf einer Web-Nachricht



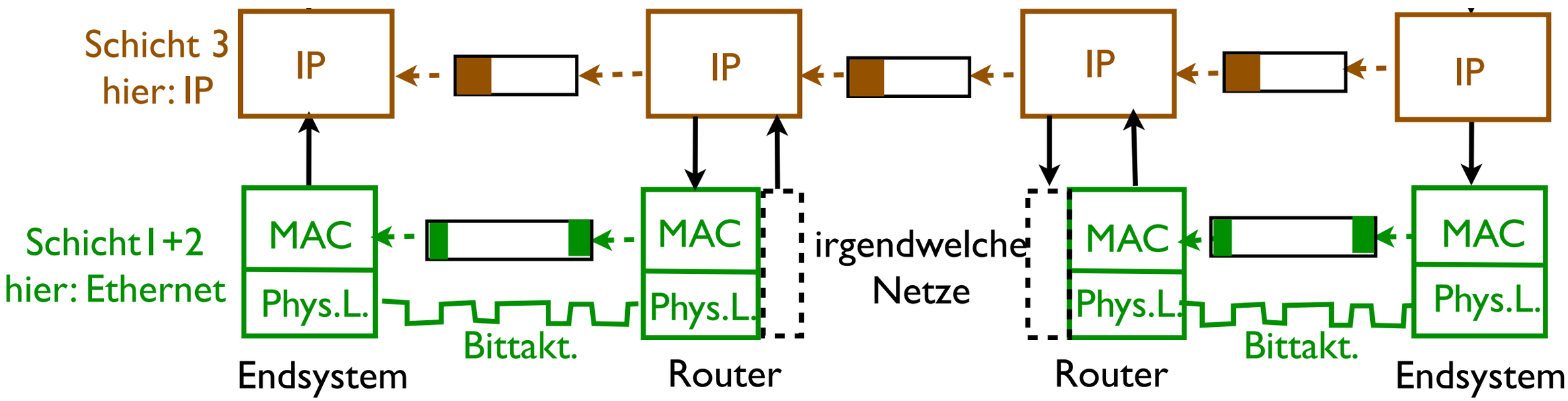
Vereinfachter Kommunikationsablauf einer Web-Nachricht





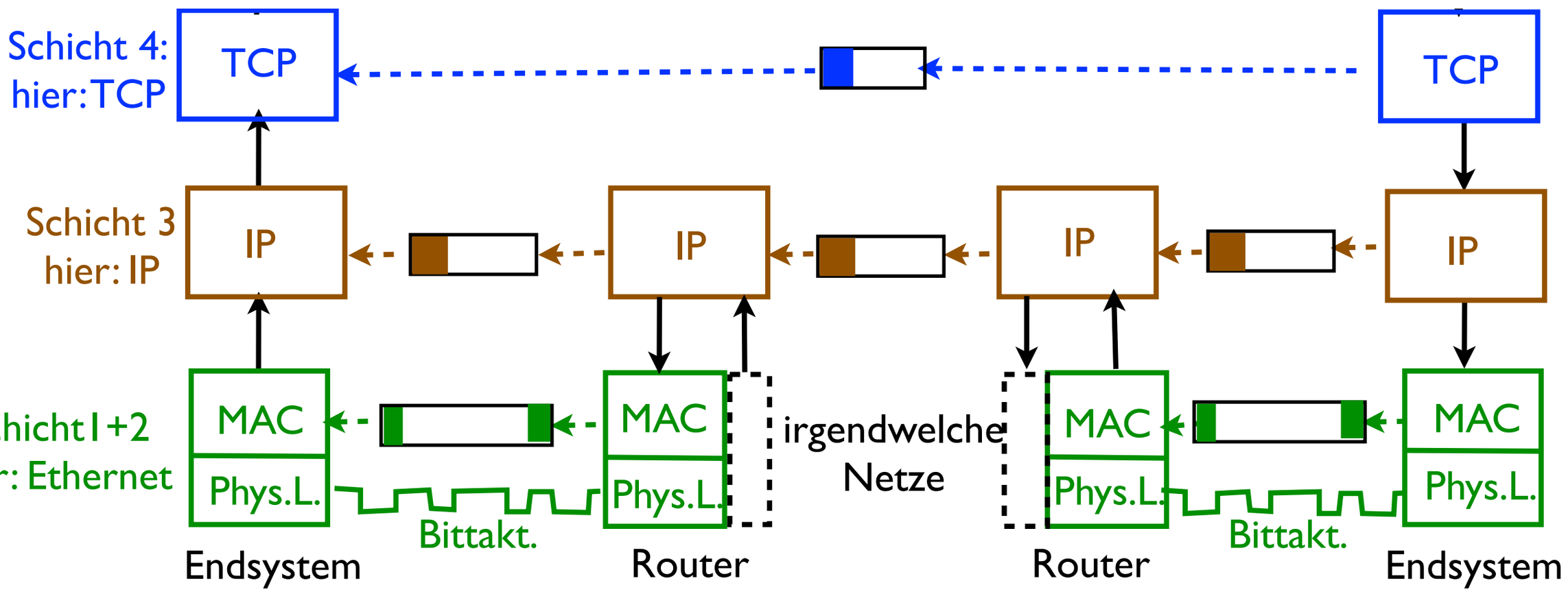
Vereinfachter Kommunikationsablauf einer Web-Nachricht

Server ← Web-Nachricht ← Client

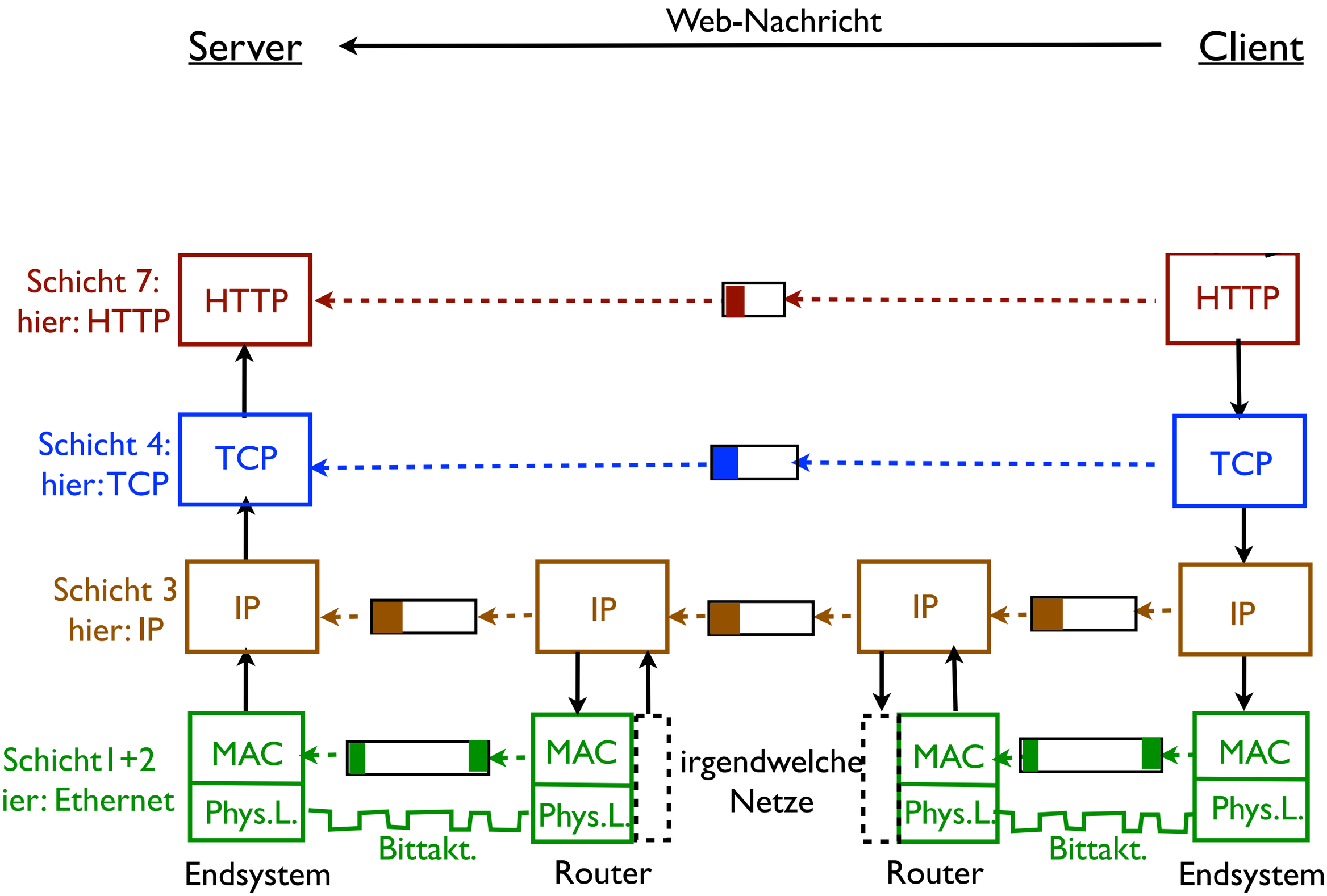


Vereinfachter Kommunikationsablauf einer Web-Nachricht

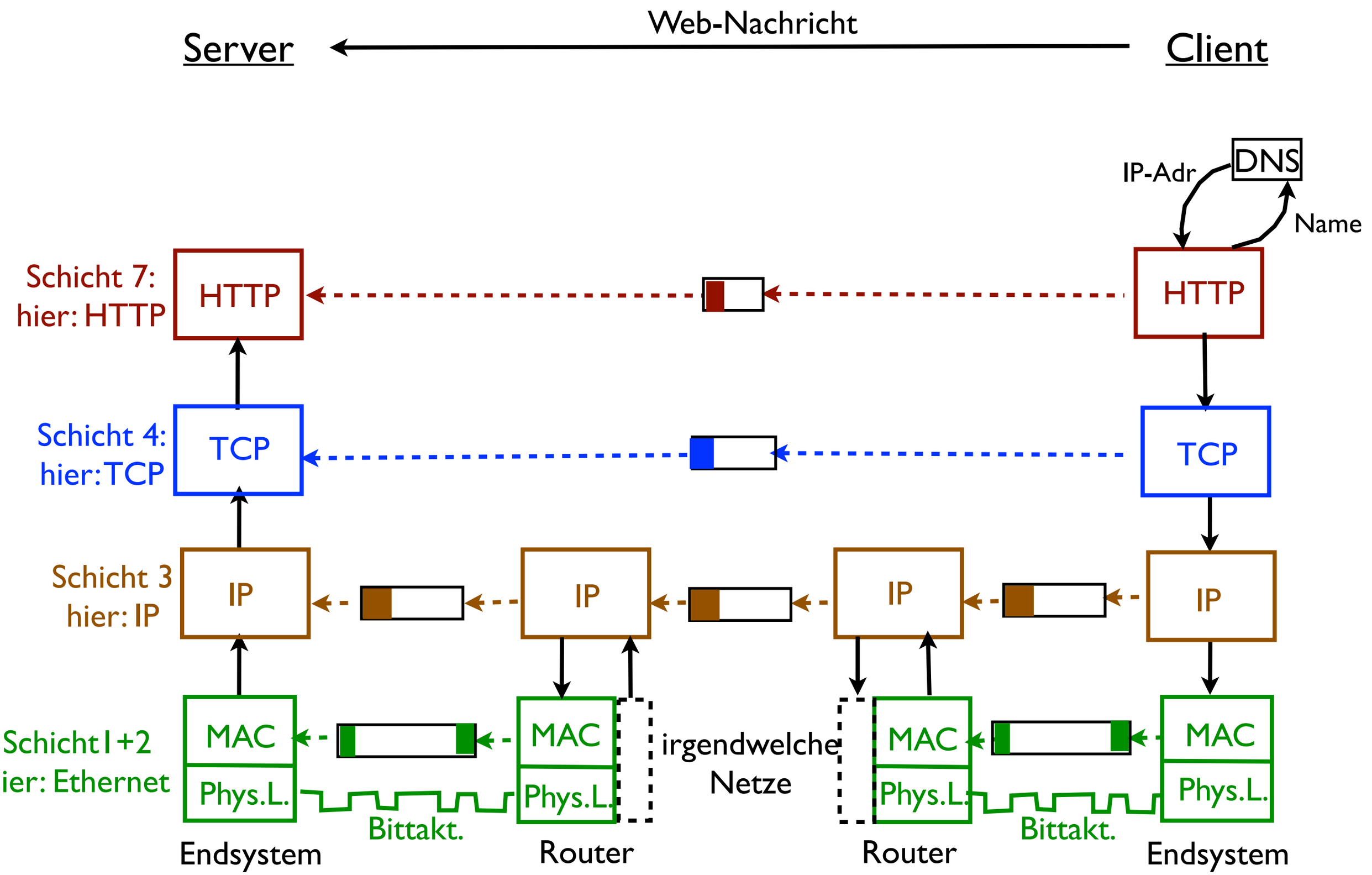
Server ← Web-Nachricht ← Client



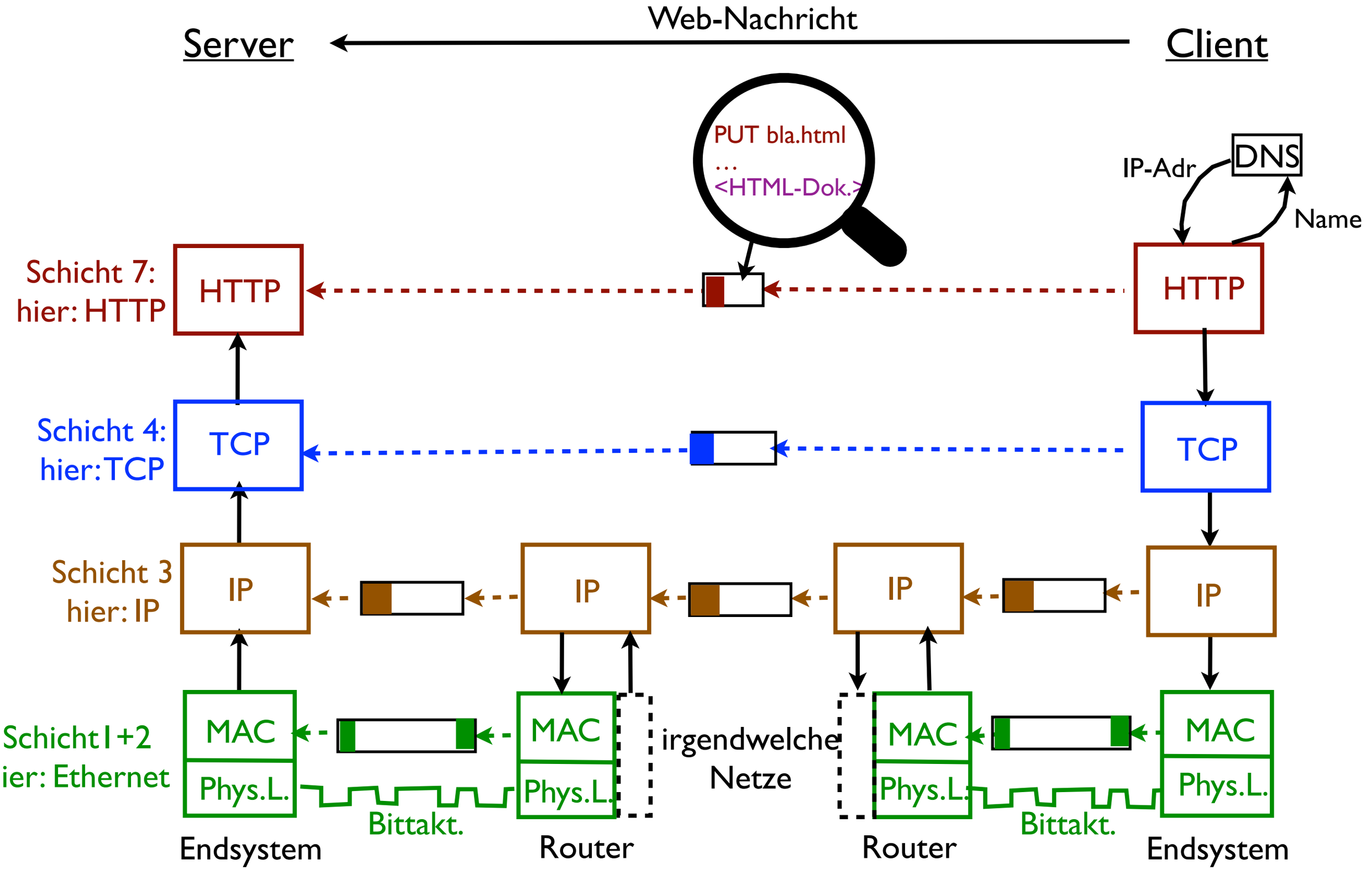
Vereinfachter Kommunikationsablauf einer Web-Nachricht



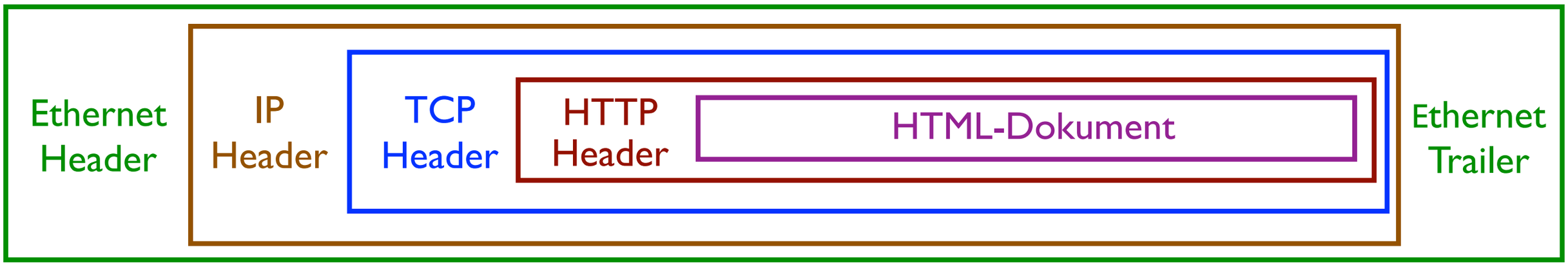
Vereinfachter Kommunikationsablauf einer Web-Nachricht



Vereinfachter Kommunikationsablauf einer Web-Nachricht



Zusammensetzung des Ethernet-Pakets in unserem Beispiel



Ethernet: Medium Access Control (MAC)

- (Aufbauend auf physischer Netztopologie:
  Heute oft Baum, früher oft Bus)
- Begrenzung von Informationseinheiten (Pakete)
  ⇒ Paketformat (Start Delimiter — End Delimiter)
- Fehlersicherung
  ⇒ 32-Bit CRC (Cyclic Redundancy Check)
  ⇒ keine Fehlerbehebung
- Adressierung (48-Bit)
  ⇒ Quelladresse und Zieladresse in jedem Paket, da verbindungslos

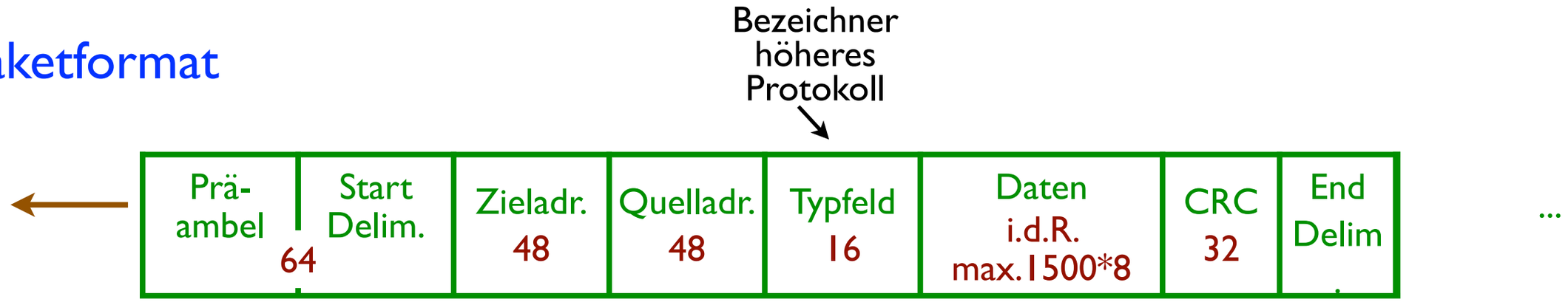
1. Bit: Adresstyp:
  = 0: einzelne Zielstation
  = 1: Gruppenadresse
48 1-Bits: Broadcast

Ethernet: Medium Access Control (MAC)

- (Aufbauend auf physischer Netztopologie:
  Heute oft Baum, früher oft Bus)
- Begrenzung von Informationseinheiten (Pakete)
  ⇒ Paketformat (Start Delimiter — End Delimiter)
- Fehlersicherung
  ⇒ 32-Bit CRC (Cyclic Redundancy Check)
  ⇒ keine Fehlerbehebung
- Adressierung (48-Bit)
  ⇒ Quelladresse und Zieladresse in jedem Paket, da verbindungslos

  1. Bit: Adresstyp:
  = 0: einzelne Zielstation
  = 1: Gruppenadresse
  48 1-Bits: Broadcast

- Paketformat



Fragen – Teil 1

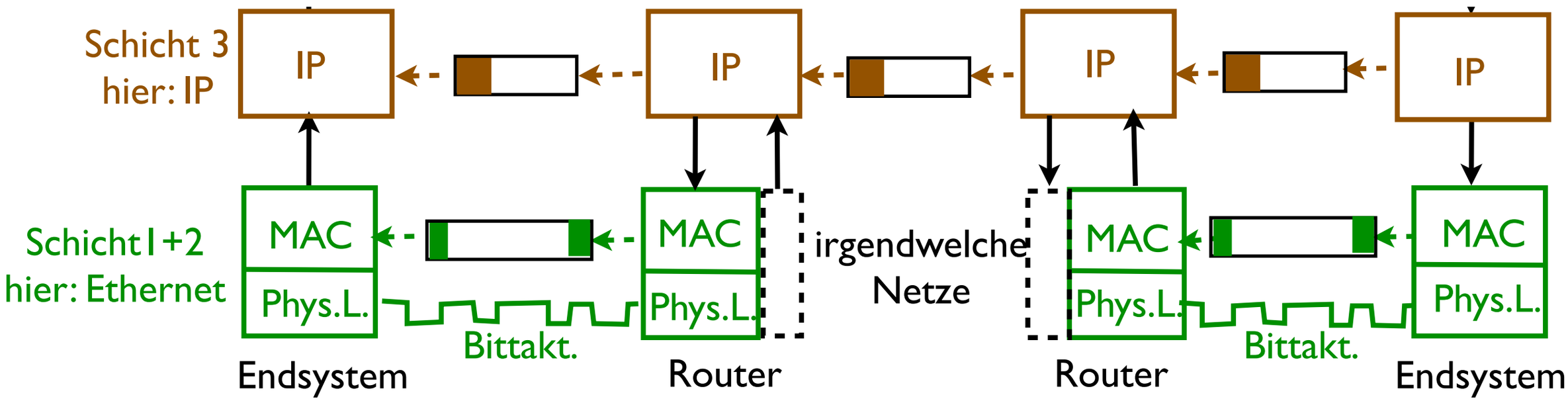
- Wie kann eine Nachricht im Grundsatz über eine Hierarchie von Protokollen versendet werden?

Teil 2:
Internet Protocol (IP)

Vereinfachter Kommunikationsablauf einer Web-Nachricht

Web-Nachricht

Server ← Client



Internet-Protokoll (IP)

IP-Dienst

- Verbindungslos ⇒ Datagramme
- keine Verzögerungs-/Durchsatzgarantien
- nicht zuverlässig

  ⇒ Daten können verfälscht werden/verlorengehen/dupliziert werden
- nicht sequenzerhaltend

⇒ nicht besser als die Dienste der unterliegenden Einzelnetze
(= Subnetze)

⇒ Best-Effort-Kommunikation

Internet-Protokoll (IP)

IP-Dienst

- Verbindungslos ⇒ Datagramme
- keine Verzögerungs-/Durchsatzgarantien
- nicht zuverlässig

  ⇒ Daten können verfälscht werden/verlorengehen/dupliziert werden
- nicht sequenzerhaltend

⇒ nicht besser als die Dienste der unterliegenden Einzelnetze (= Subnetze)

⇒ Best-Effort-Kommunikation

- Wesentlicher Dienst: Netzübergreifende Adressierung
- Spätere Erweiterungen/Zusatzprotokolle...

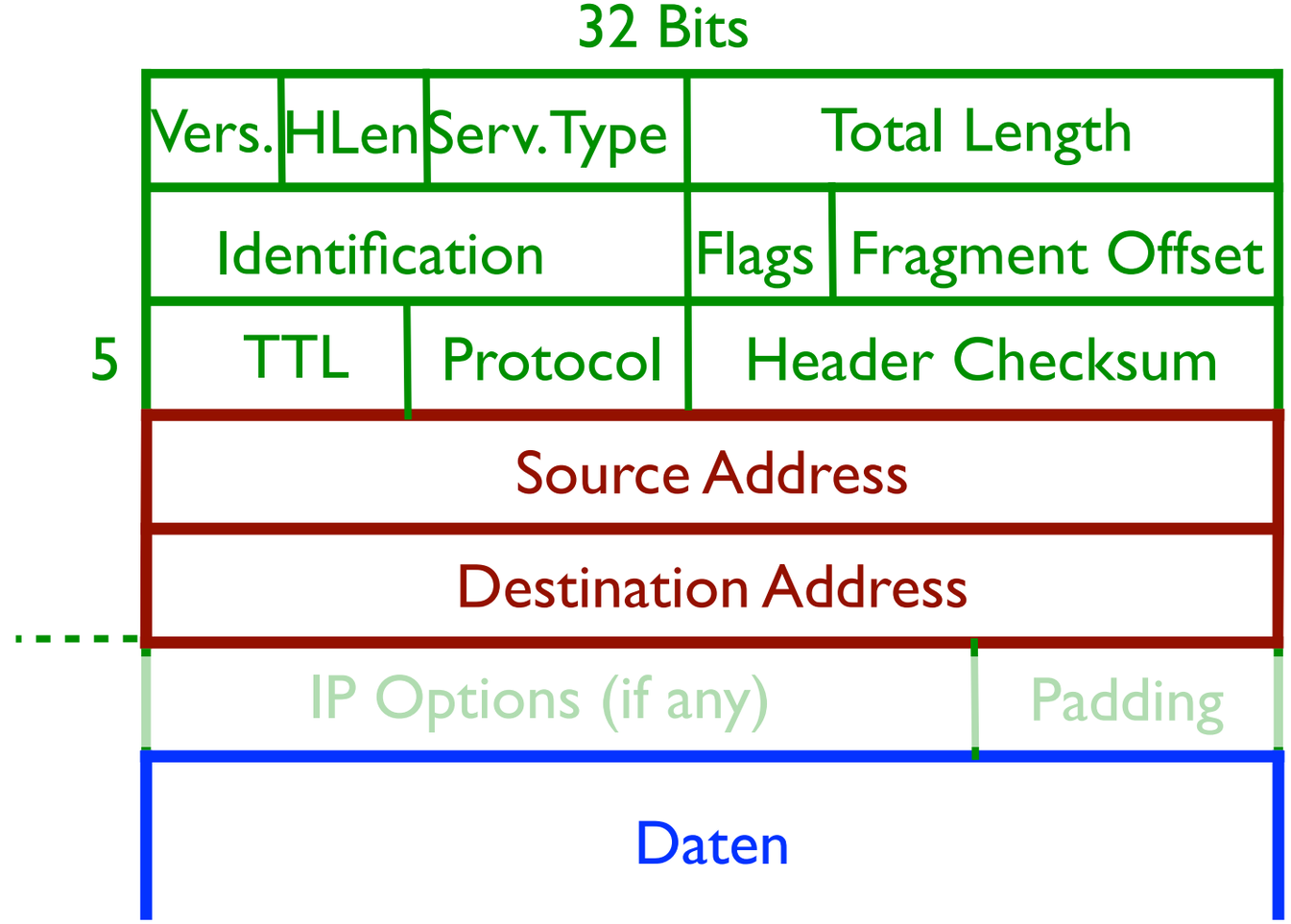
IPv4-Header

Idee:

- IP-Header enthält (Internet-weit) eindeutige Adresse des Sender- und Empfängersystems

Zwei Adressteile:

- Netzbezeichnung
- Hostadresse innerhalb des Netzes



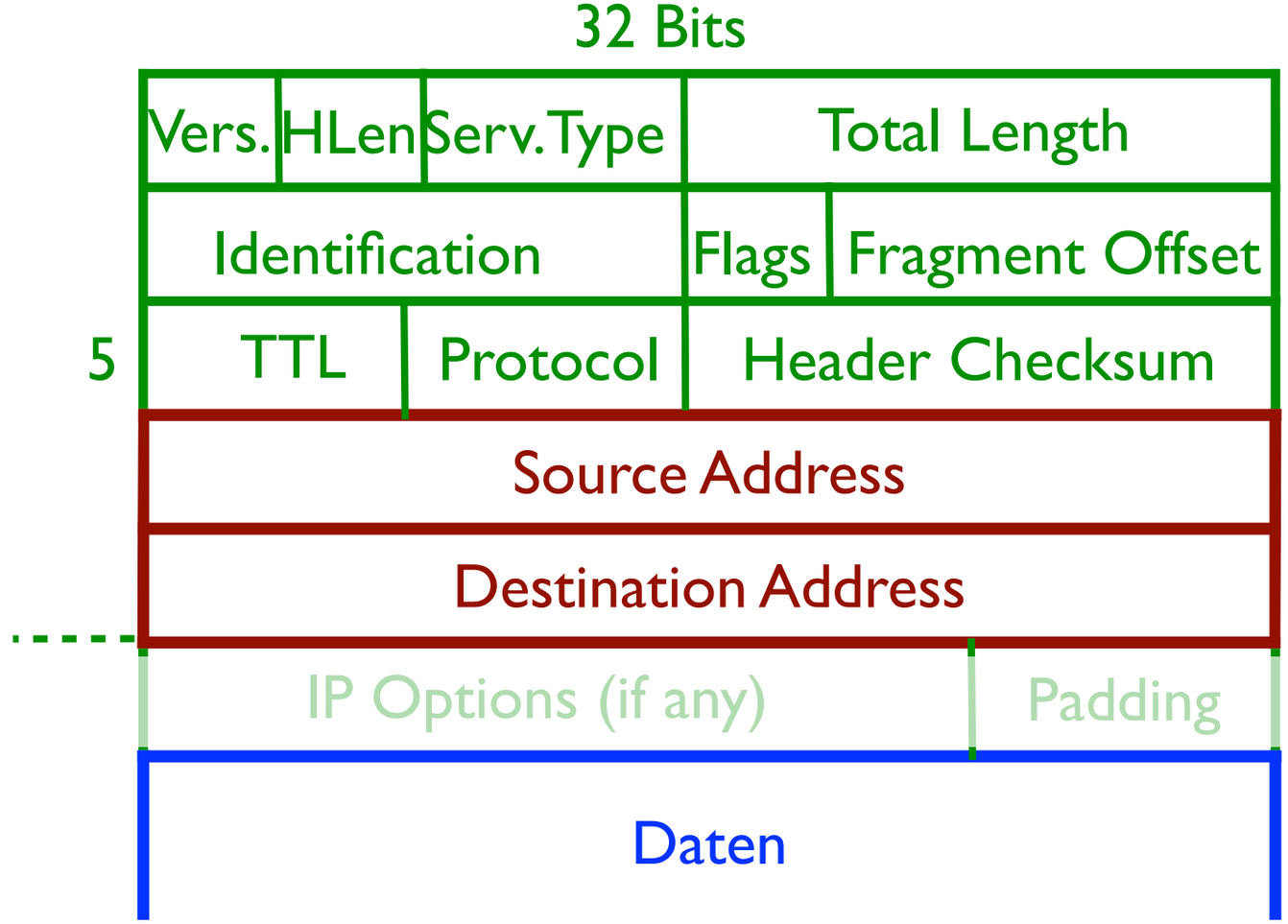
IPv4-Header

Idee:

- IP-Header enthält (Internet-weit) eindeutige Adresse des Sender- und Empfängersystems

Zwei Adressteile:

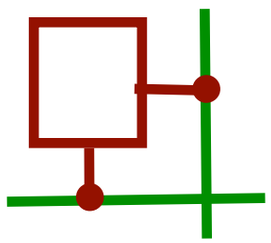
- Netzbezeichnung
- Hostadresse innerhalb des Netzes



- Genauer: IP-Adresse benennt „Netz-Interface“

  ⇒ Ein Host kann mehrere Adressen haben

- Vergleich: Straße vs. Hausnummer



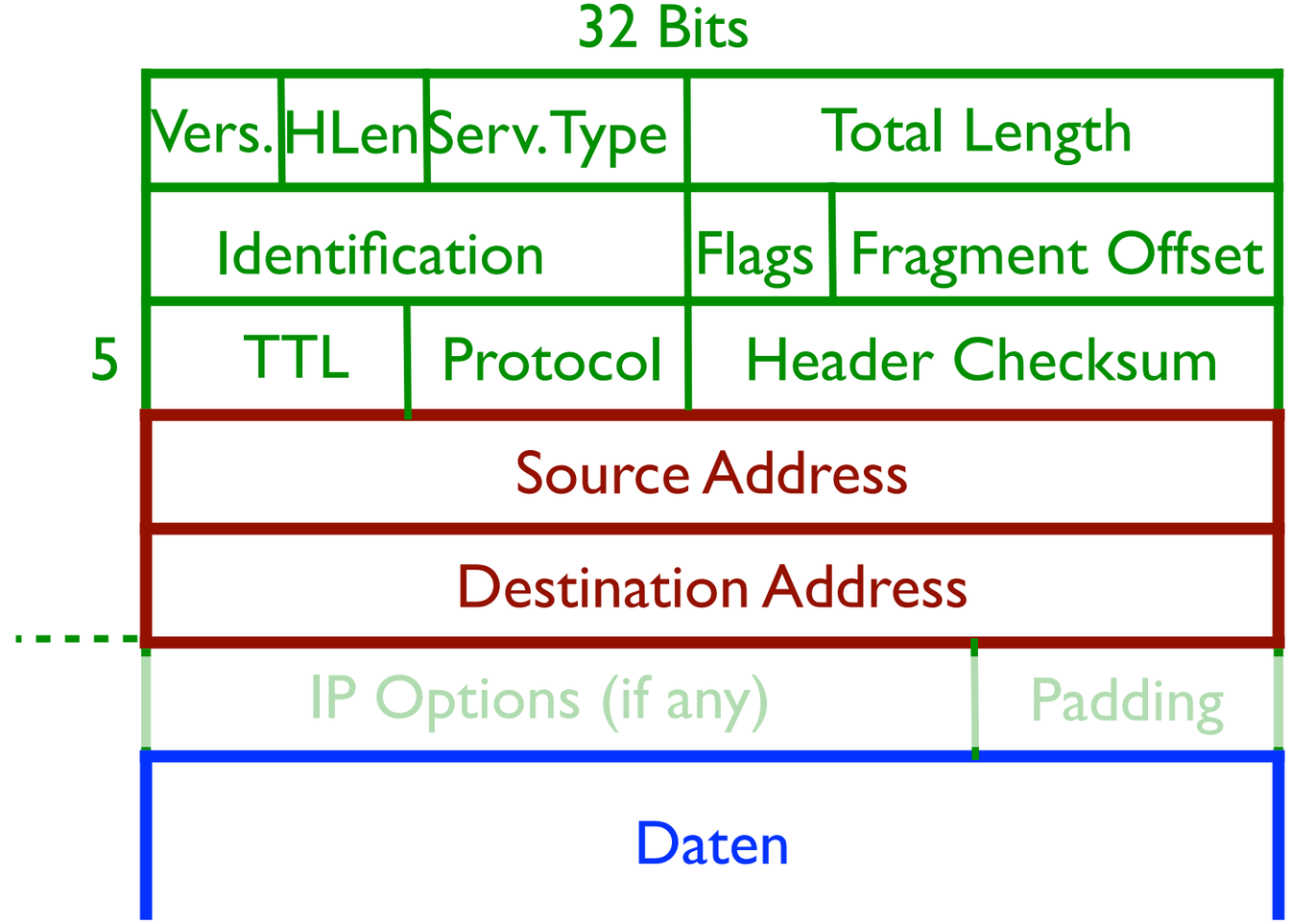
IPv4-Header

Idee:

- IP-Header enthält (Internet-weit) eindeutige Adresse des Sender- und Empfängersystems

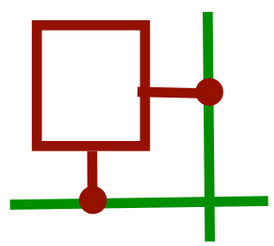
Zwei Adressteile:

- Netzbezeichnung
- Hostadresse innerhalb des Netzes



- Genauer: IP-Adresse benennt „Netz-Interface“

  ⇒ Ein Host kann mehrere Adressen haben



- Vergleich: Straße vs. Hausnummer

- Netze unterschiedlich groß

  ⇒ Unterschiedliche Aufteilung des Nummernraums

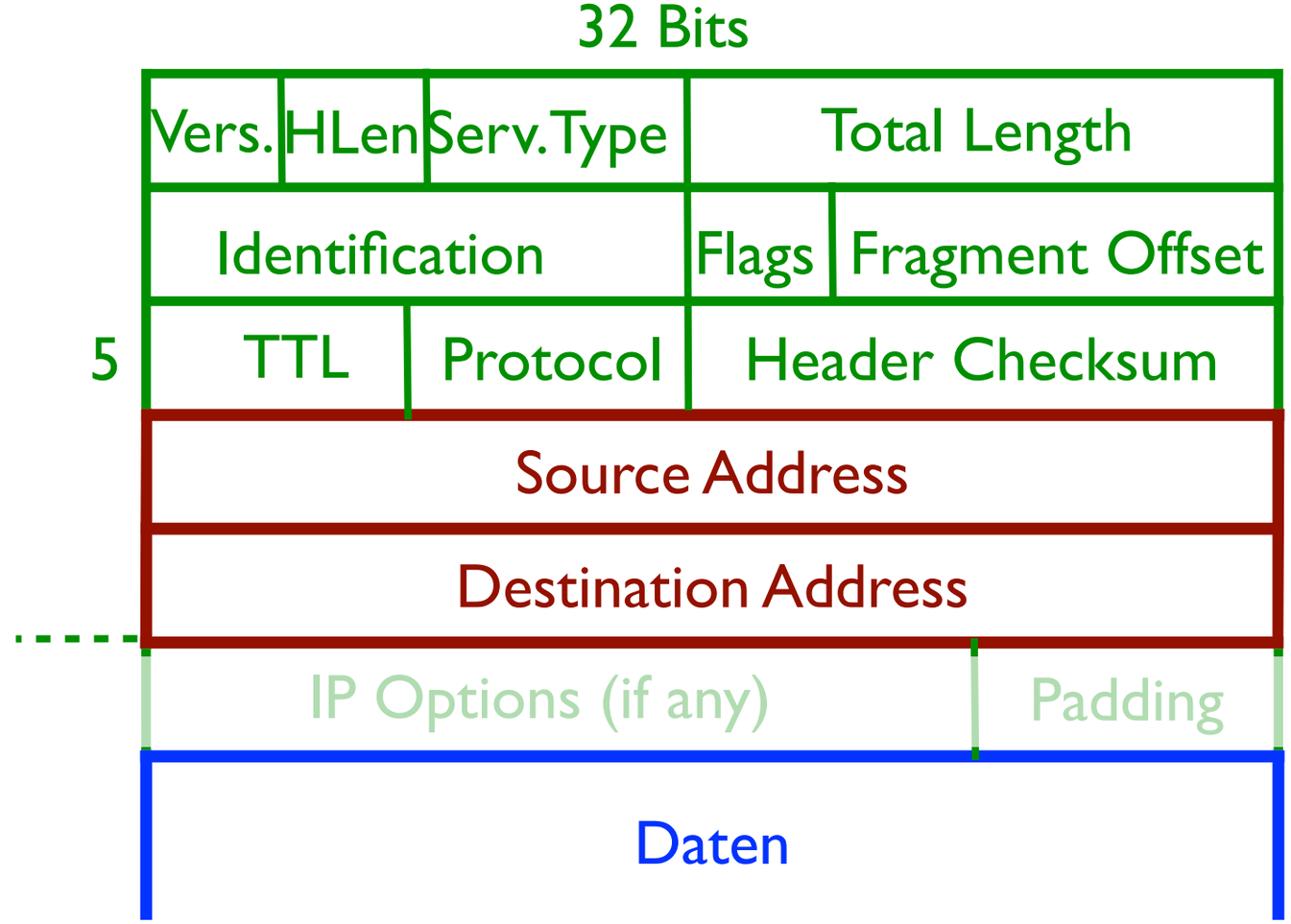
IPv4-Header

Idee:

- IP-Header enthält (Internet-weit) eindeutige Adresse des Sender- und Empfängersystems

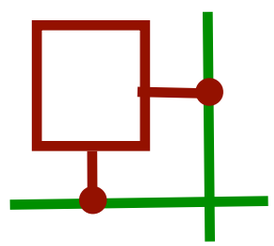
Zwei Adressteile:

- Netzbezeichnung
- Hostadresse innerhalb des Netzes



- Genauer: IP-Adresse benennt „Netz-Interface“

  ⇒ Ein Host kann mehrere Adressen haben



- Vergleich: Straße vs. Hausnummer

- Netze unterschiedlich groß

  ⇒ Unterschiedliche Aufteilung des Nummernraums

- Host-ID = 0 ⇒ Adresse des Netzes
  Host-ID = max ⇒ alle Hosts im Netz (z.B. 255 bei 8-Bit Host-Anteil)

- Host-ID = 0 ⇒ Adresse des Netzes
  Host-ID = max ⇒ alle Hosts im Netz (z.B. 255 bei 8-Bit Host-Anteil)

- Schreibweise von Internet-Adressen (im IP-Protokoll selbst nur unstrukturierte 32-Bit-Nummer):

  `Byte1.Byte2.Byte3.Byte4` (dezimal)

  z.B. `134.102.218.42`

- Host-ID = 0 ⇒ Adresse des Netzes
  Host-ID = max ⇒ alle Hosts im Netz (z.B. 255 bei 8-Bit Host-Anteil)

- Schreibweise von Internet-Adressen (im IP-Protokoll selbst nur unstrukturierte 32-Bit-Nummer):

  `Byte1.Byte2.Byte3.Byte4` (dezimal)

  z.B. `134.102.218.42`

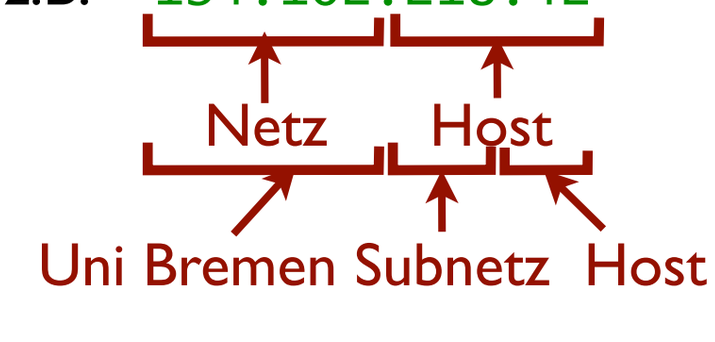
(ggf. Subnetz: lokale Unterstruktur des Netzes)

- Host-ID = 0 ⇒ Adresse des Netzes
  Host-ID = max ⇒ alle Hosts im Netz (z.B. 255 bei 8-Bit Host-Anteil)

- Schreibweise von Internet-Adressen (im IP-Protokoll selbst nur unstrukturierte 32-Bit-Nummer):

  `Byte1.Byte2.Byte3.Byte4` (dezimal)

  z.B. `134.102.218.42`



(ggf. Subnetz: lokale Unterstruktur des Netzes)

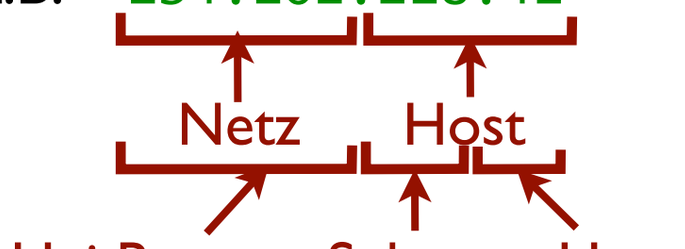
- Netzpräfix-Notation (in IP-Protokoll nicht sichtbar): z.B. `135.16.0.0/12`

- Host-ID = 0 ⇒ Adresse des Netzes
  Host-ID = max ⇒ alle Hosts im Netz (z.B. 255 bei 8-Bit Host-Anteil)

- Schreibweise von Internet-Adressen (im IP-Protokoll selbst nur unstrukturierte 32-Bit-Nummer):

  `Byte1.Byte2.Byte3.Byte4` (dezimal)

  z.B. `134.102.218.42`



(ggf. Subnetz: lokale Unterstruktur des Netzes)

- Netzpräfix-Notation (in IP-Protokoll nicht sichtbar): z.B. `135.16.0.0/12`

- Adressbereich für „private“ Adressen
  ⇒ Verwendung in NATs

  `10.0.0.0/8`
  `172.16.0.0/12`
  `192.168.0.0/16`

- Falls Empfänger in anderem Netz

  ⇒ keine direkte Versendung möglich

  ⇒ Ansprechen eines Routers mit Hilfe des unterliegenden netzinternen Protokolls (z.B. Ethernet) und netzinterner Adresse des Routers

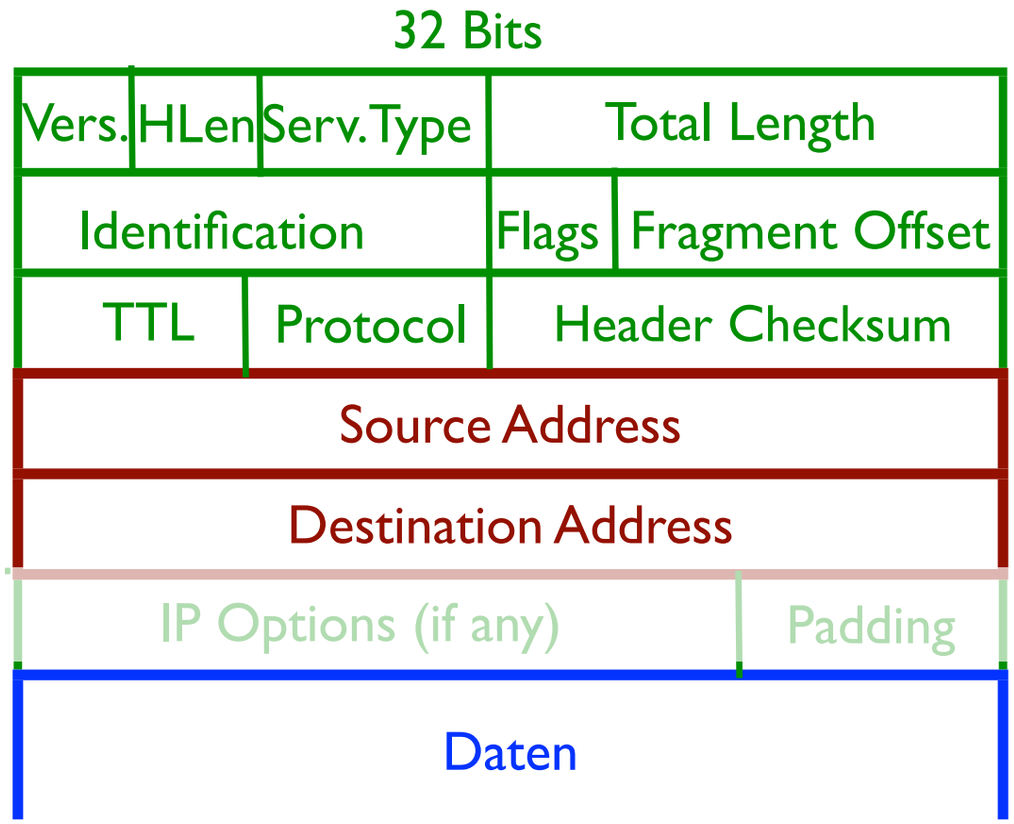
- Router benötigt dazu Routing-Tabelle:
  - Angabe von Netzpräfixen und zugehörigem Output-Port
  - I.d.R. erzeugt/aktualisiert durch Routing-Protokoll
  - Spezifischster Eintrag „gewinnt“

sehr vereinfacht

Netzpräfix	Output-Port
237.165.96.0/20	1
237.165.0.0/18	2
237.165.64.0/20	3
237.165.112.0/24	3
237.164.0.0/16	4
237.165.113.128/26	4
DEFAULT	5

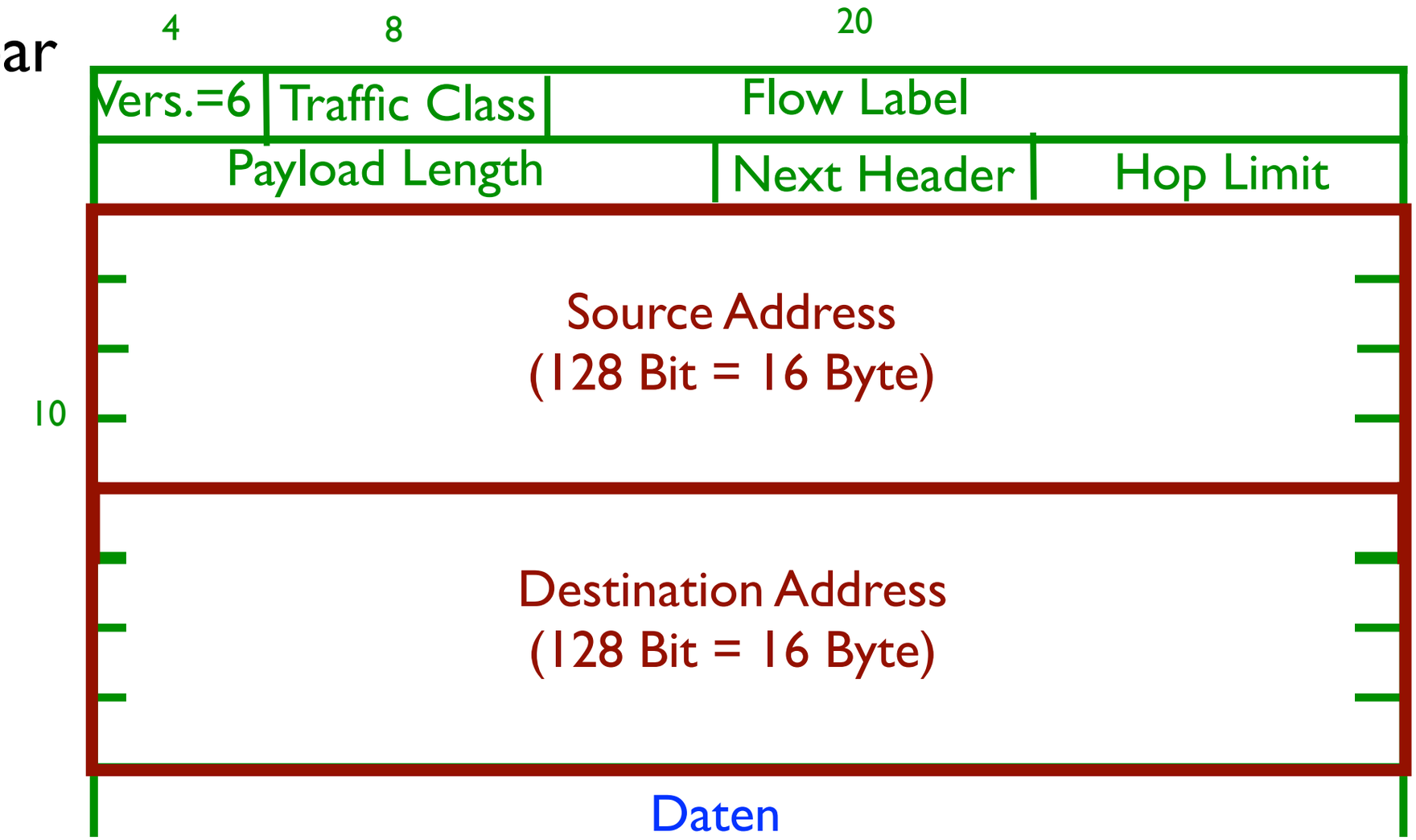
IPv4-Header (Fortsetzung)

- Vers: Version 4
- HLen/Total Length: Längenangaben
- Header Checksum: Prüfsumme (nur über Header)
- Identification: Datagrammbezeichner
- Flags:
  - Darf fragmentiert werden?
  - Anfang/Ende des Datagramms....
- Fragment Offset: Welches Fragment?
- Protokoll: ID des nächsthöheren Protokolls (z.B. TCP)
- Time to live (TTL): Wie lange „lebt“ Datagramm maximal im Netz (# hops)
- Service Type: u.U. „Dienstgüteklasse“
  ⇒ oft nur „Hint“



IPv6

- Erforderlich, weil IPv4-Adressen ausgehen
- Lange Übergangsphase
- Durchbruch absehbar
- Header-Format:



- Traffic Class: Für verschiedene Dienstgüteklassen ... (vgl. IPv4 Service Type)
- Payload Length Länge der enthaltenen Daten (vgl. IPv4 Total Length)
- Hop Limit: Hop Count (vgl. IPv4 TTL)
- Next Header: „Höheres“ Protokoll (vgl. IPv4 Protocol)
- Flow Label: Strombezeichner

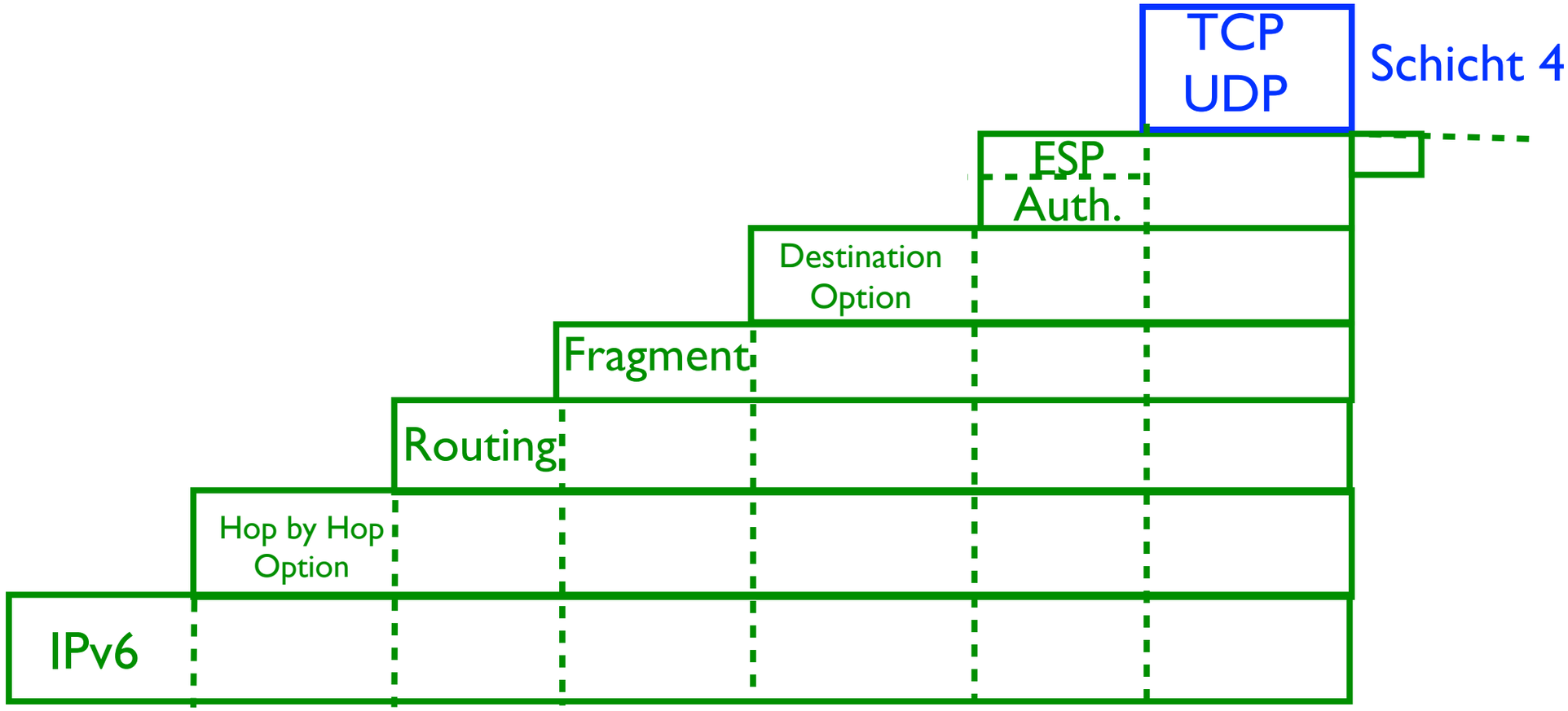
⇒ einige IPv4-Funktionalitäten ausgelagert in optionale weitere Subschichten/Header

Header-Hierarchie (auch für IPv4 nutzbar)

- Alle „Optionen“ in optionale weitere „Protokoll-/Header-Schichten“ ausgelagert

  ⇒ nur nutzen, was aktuell gebraucht wird

  ⇒ nur anschauen, was interessiert

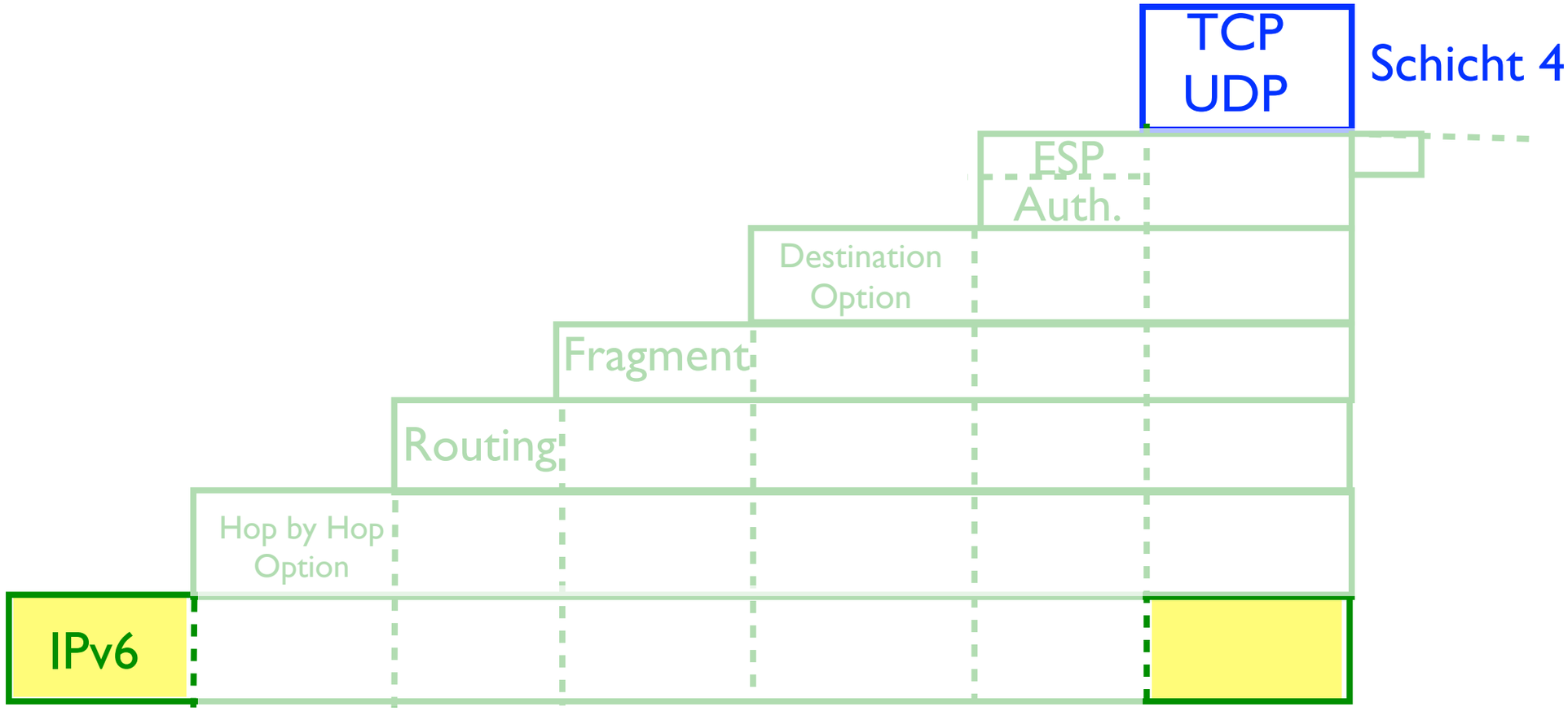


Header-Hierarchie (auch für IPv4 nutzbar)

- Alle „Optionen“ in optionale weitere „Protokoll-/Header-Schichten“ ausgelagert

  ⇒ nur nutzen, was aktuell gebraucht wird

  ⇒ nur anschauen, was interessiert

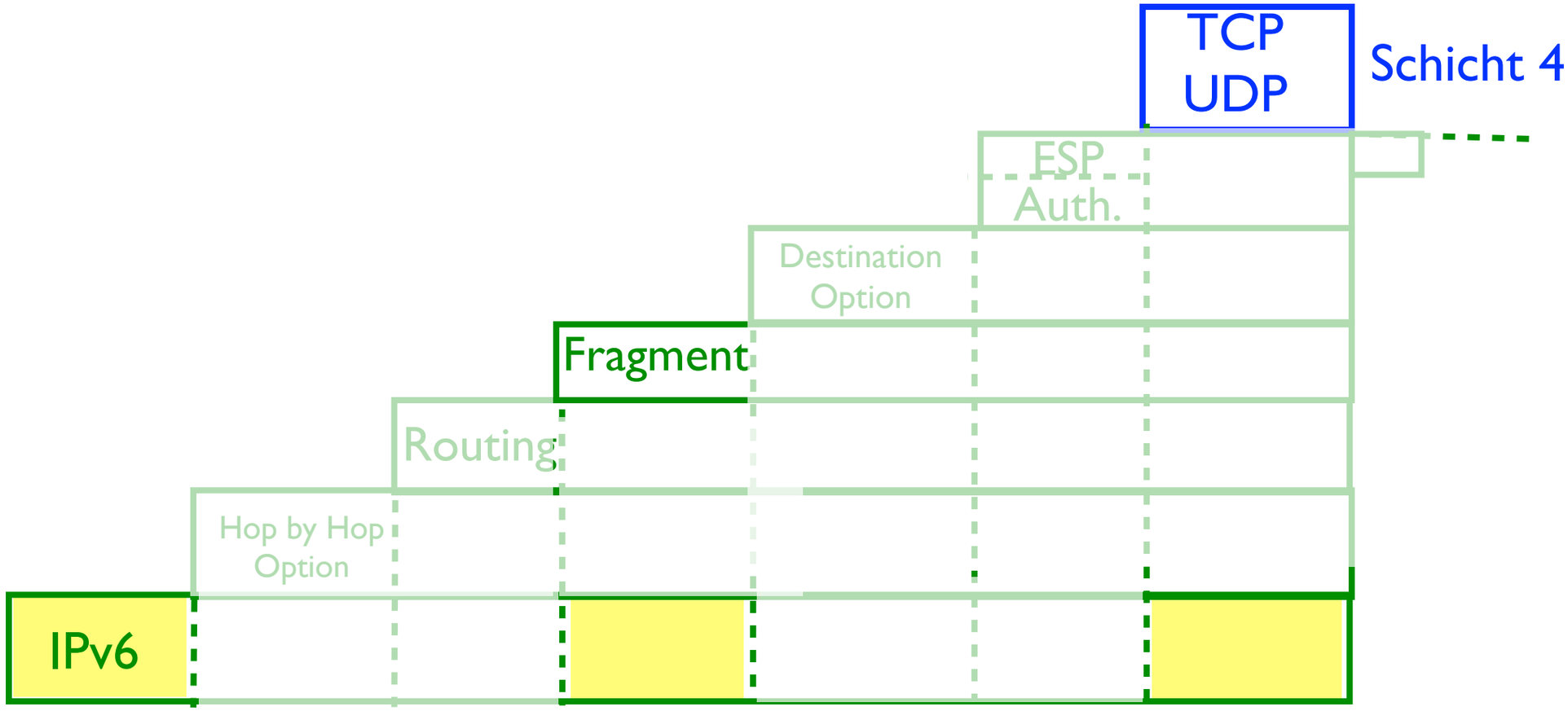


Header-Hierarchie (auch für IPv4 nutzbar)

- Alle „Optionen“ in optionale weitere „Protokoll-/Header-Schichten“ ausgelagert

  ⇒ nur nutzen, was aktuell gebraucht wird

  ⇒ nur anschauen, was interessiert



IPv6-Adressarchitektur

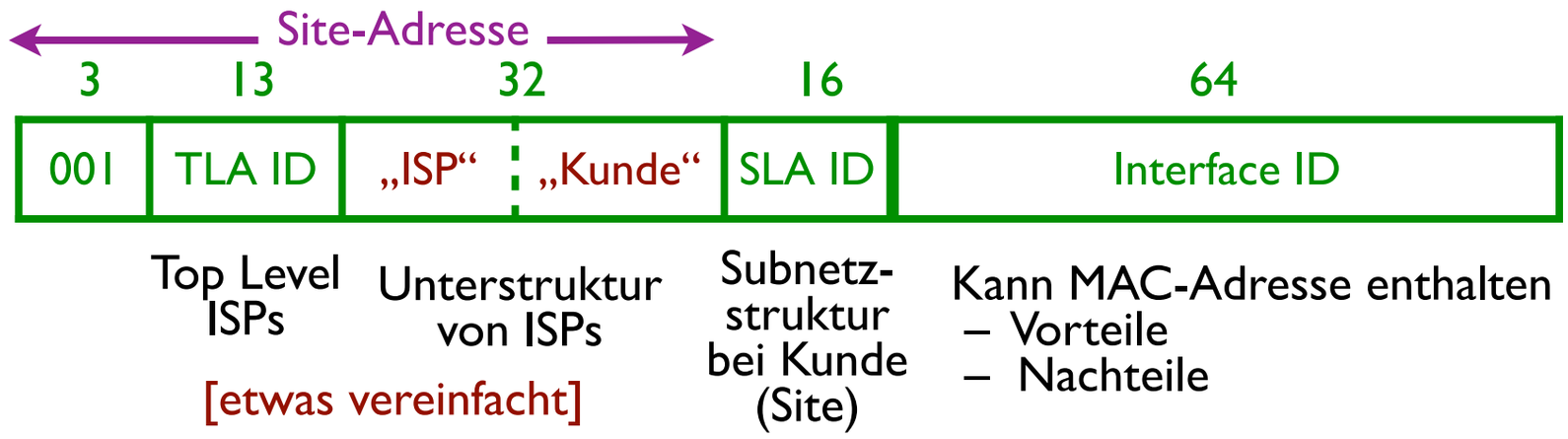
- Adresslänge 16 Bytes (statt 4 Bytes)
- Anders notiert: als 8 16-Bit-Hexzahlen
  (Notation nicht Protokollbestandteil)

  Beispiel: fedc:ba98:7654:3210:fedc:ba98:7654:3210
- Abkürzungen: z.B. ::1 (Loopback-Adresse)
- Ein Host kann mehrere (IPv4/IPv6-)Adressen haben

IPv6-Adressarchitektur

- Adresslänge 16 Bytes (statt 4 Bytes)
- Anders notiert: als 8 16-Bit-Hexzahlen
  (Notation nicht Protokollbestandteil)

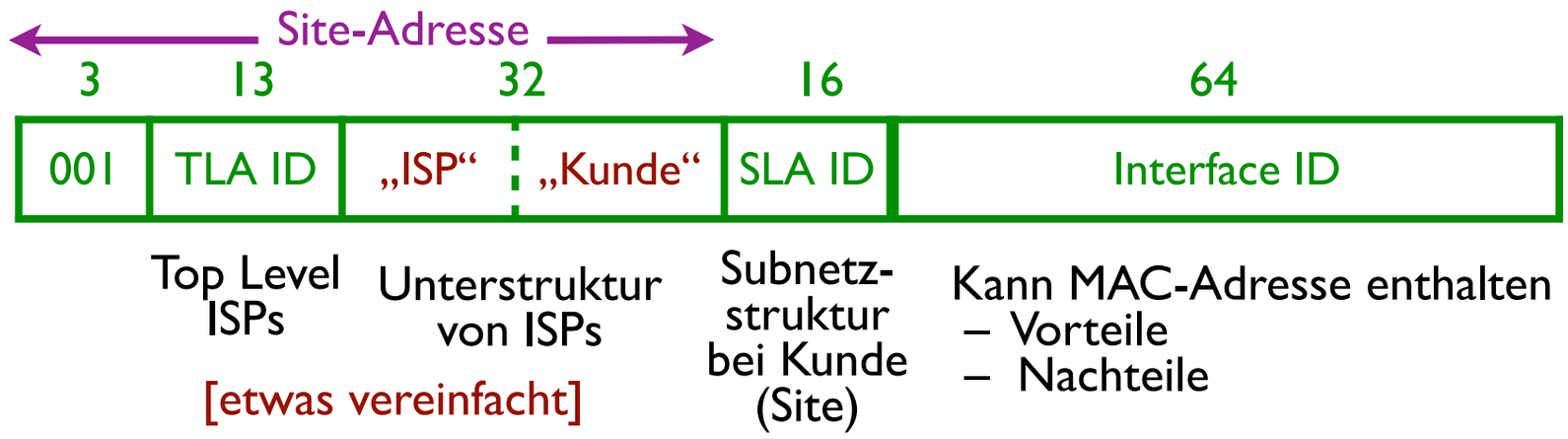
  Beispiel: fedc:ba98:7654:3210:fedc:ba98:7654:3210
- Abkürzungen: z.B. ::1 (Loopback-Adresse)
- Ein Host kann mehrere (IPv4/IPv6-)Adressen haben
- Verschiedene Adressarten ⇒ verschiedene Präfixe, z.B.
  - Global Unicast: 001



IPv6-Adressarchitektur

- Adresslänge 16 Bytes (statt 4 Bytes)
- Anders notiert: als 8 16-Bit-Hexzahlen
  (Notation nicht Protokollbestandteil)

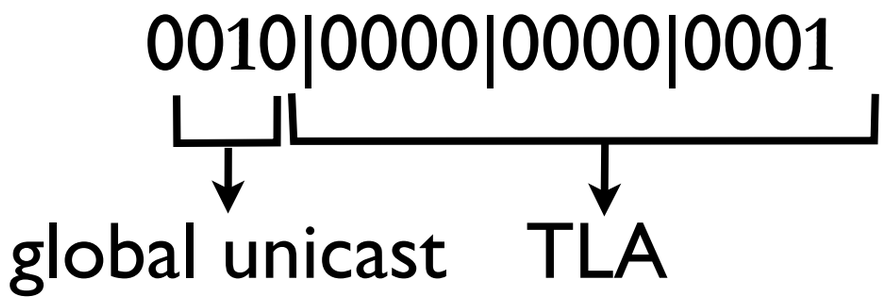
  Beispiel: fedc:ba98:7654:3210:fedc:ba98:7654:3210
- Abkürzungen: z.B. ::1 (Loopback-Adresse)
- Ein Host kann mehrere (IPv4/IPv6-)Adressen haben
- Verschiedene Adressarten ⇒ verschiedene Präfixe, z.B.
  - Global Unicast: 001



  - Link-Local Unicast: 1111 1110 10 ….

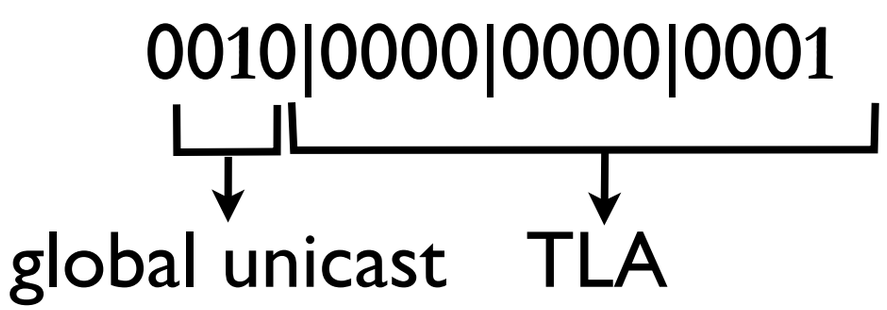
Beispiel für eine IPv6-Adresse

2001::/16



Beispiel für eine IPv6-Adresse

2001::/16

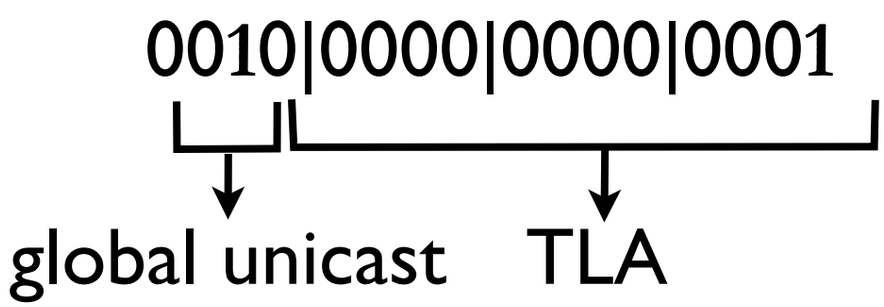


2001:0638::/32 DFN-Verein

2001:638:708::/48 Uni Bremen

Beispiel für eine IPv6-Adresse

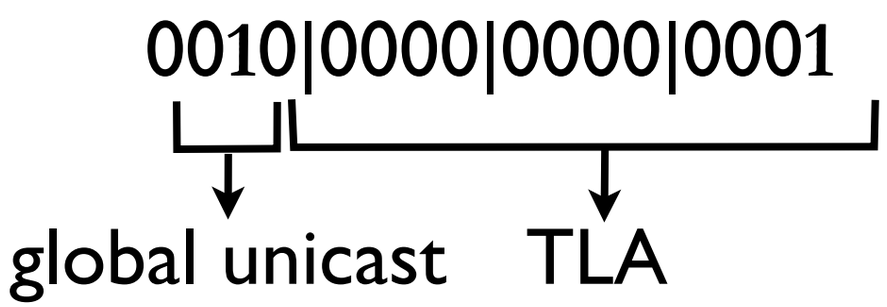
2001::/16



2001:0638::/32	DFN-Verein
2001:638:708::/48	Uni Bremen
2001:638:708:3000::/52	FB3
2001:638:708:3000::/56	FB3-Netze, die auch IPv4-Netze sind
2001:638:708:30da::/64	Netz 0xda = 218 (≙ 134.102.218.0/24)

Beispiel für eine IPv6-Adresse

2001::/16



2001:0638::/32 DFN-Verein

2001:638:708::/48 Uni Bremen

2001:638:708:3000::/52 FB3

2001:638:708:3000::/56 FB3-Netze, die auch IPv4-Netze sind

2001:638:708:30da::/64 Netz 0xda = 218 (≙ 134.102.218.0/24)

2001:638:708:30da:020e:0cff:fe37:4156 (Host „restoration“)

- Hostadresse kann (Variante der) MAC-Adresse sein (Vor-/Nachteile)

Kleine Aufgabe

- Gegeben sei folgende/s IPv4-Adresse/Subnetz:

  192.160.68.250/20

- Welche binäre Adresse hat das Netz, welche der Host innerhalb des Netzes?
- Was steht in diesem Fall im Adressfeld innerhalb des IP-Pakets?

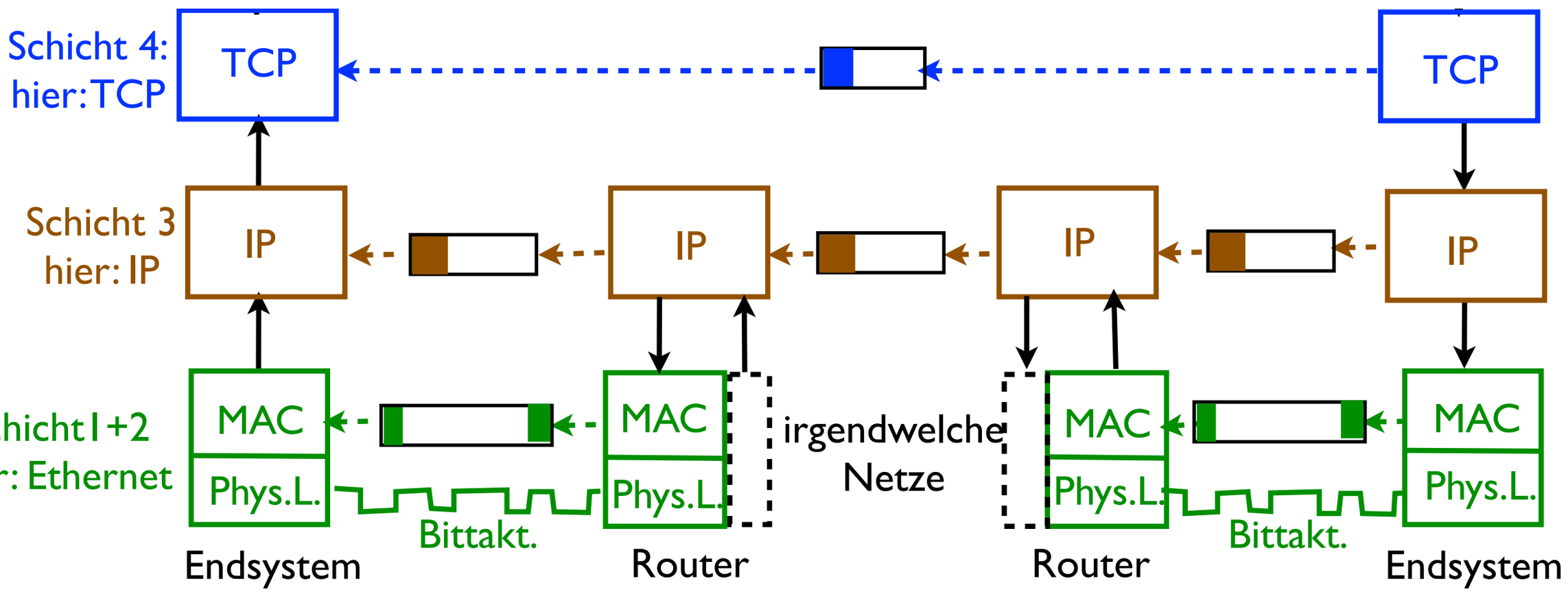
Fragen – Teil 2

- Aus welchen Teilen besteht eine IP-Adresse?
- Worin unterscheiden sich IPv4 und IPv6?
- Wozu enthalten IP-Pakete ein TTL-Feld?

Teil 3:
Transmission Control Protocol (TCP)
und Domain Name Service (DNS)

Vereinfachter Kommunikationsablauf einer Web-Nachricht

Server ← Web-Nachricht ← Client



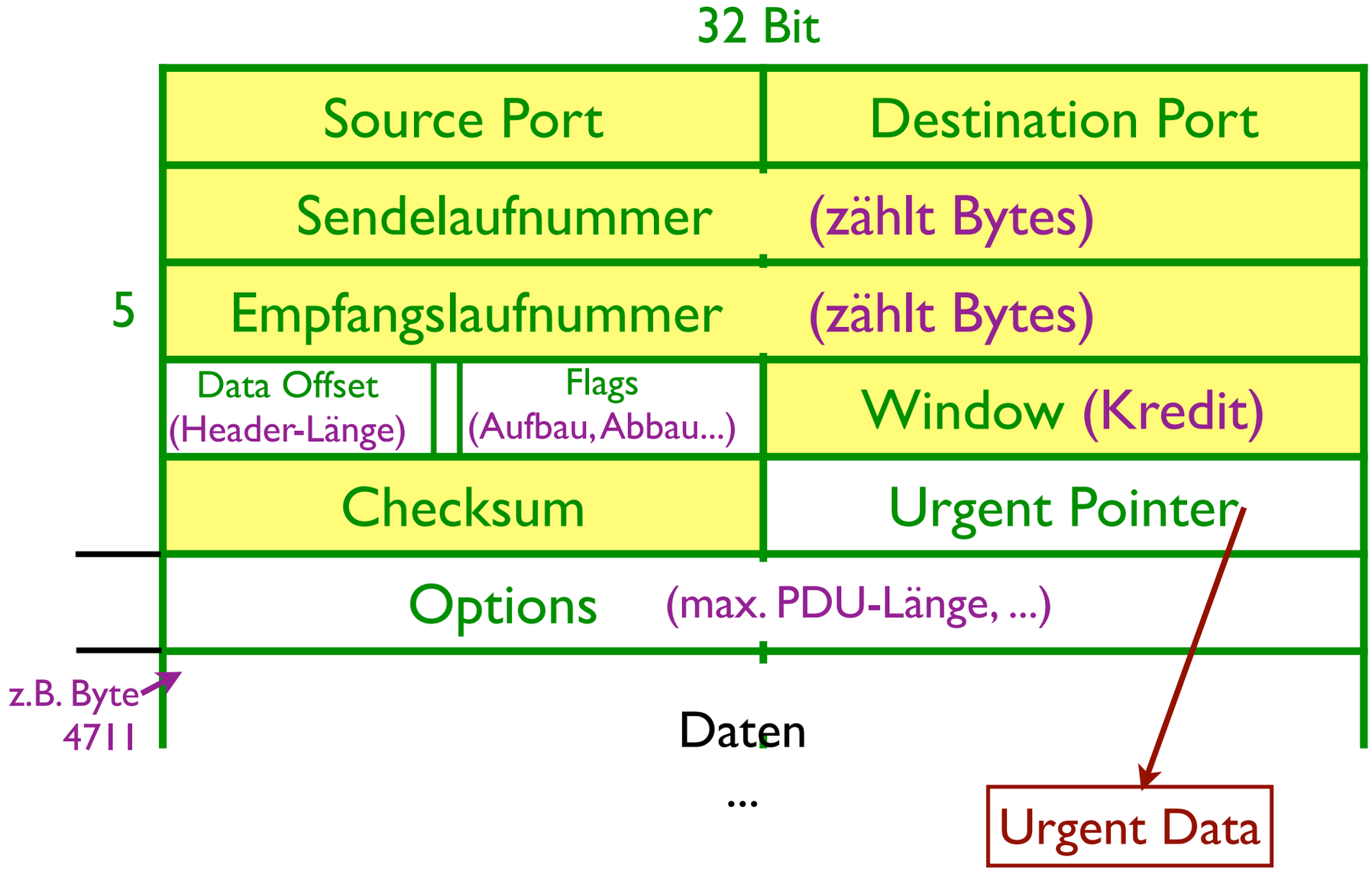
TCP (Transmission Control Protocol)

- Transportprotokoll oberhalb von IP
- verbindungsorientiert
- Adressierung von Anwendungen
  (16-Bit-Portnummern)

TCP (Transmission Control Protocol)

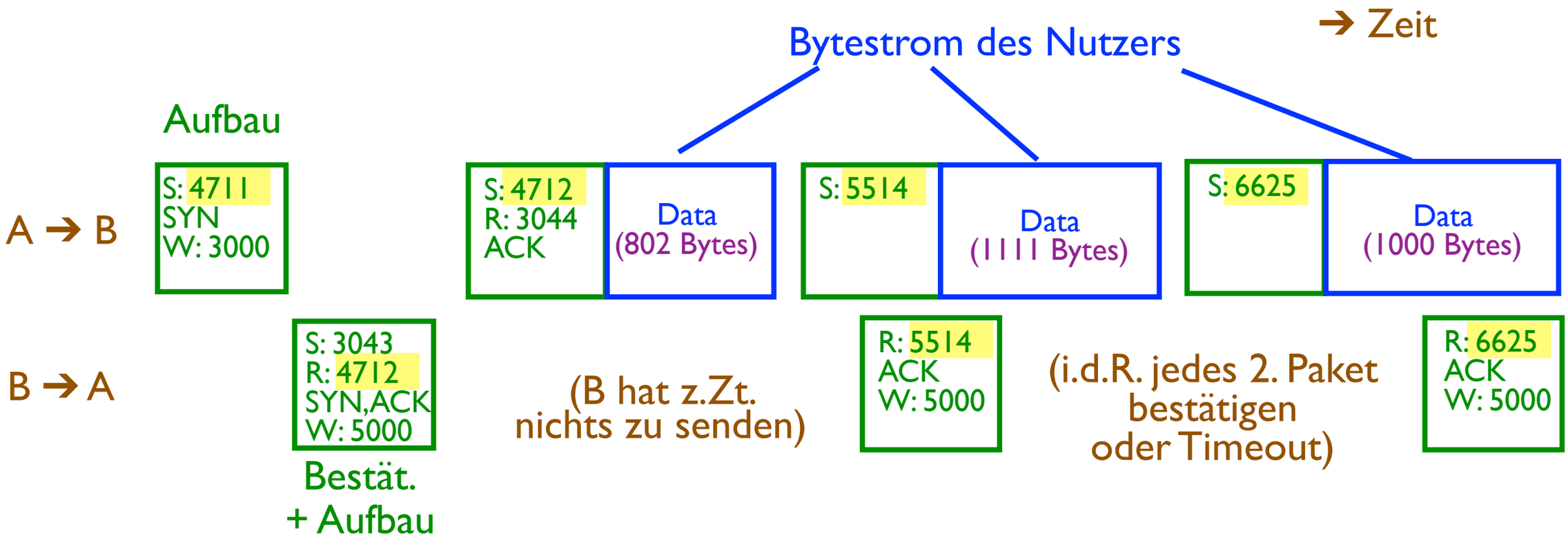
- Transportprotokoll oberhalb von IP
- verbindungsorientiert
- Adressierung von Anwendungen (16-Bit-Portnummern)
- Qualitätsverbesserung
  - Fehlererkennung (Prüfsumme)
  - Fehlerbehebung (Wiederholung)
  - Flusskontrolle (Überlastschutz des Empfängers)
  - Staukontrolle (Überlastschutz der Netzknoten, z.B. Router)
  - Wiederherstellung der Reihenfolge
  - Mehrere Transportströme unterscheidbar (Art Multiplexing)

TCP-Header (i.d.R. 20 Bytes)



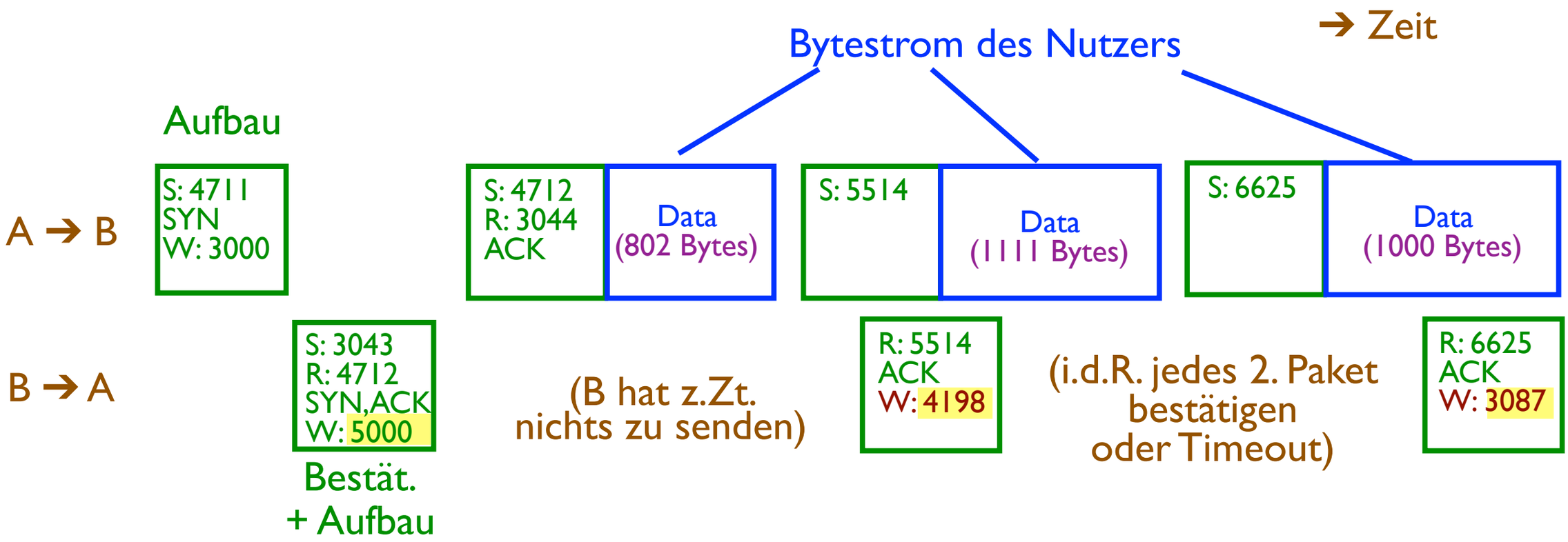
- Flags:
  - SYN Aufbau
  - FIN Abbau
  - RST auch: Abbruch / Protokollfehler
  - ACK Empfangslaufnummer gilt
  - URG Urgent Pointer gilt
- Window: Wieviele Bytes können noch empfangen werden (relativ zu enthaltenen ACK-Nr) ⇒ Flusskontrolle

Bestätigungsmechanismus



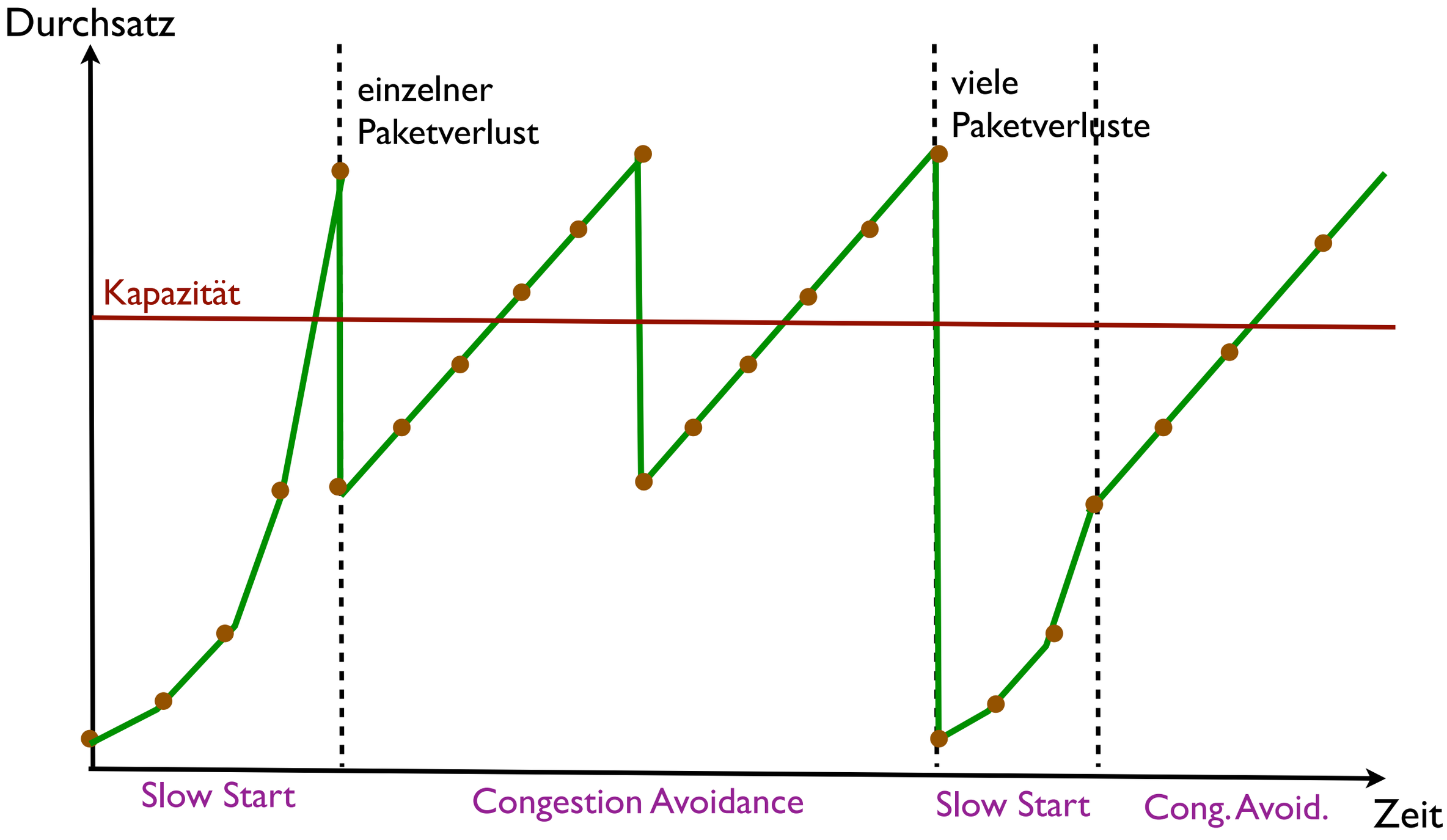
- Aufbau (SYN): Virtuelles 0. Byte
- bei beliebiger Nummer beginnen (+ Sende-Portnummer eine Weile einfrieren)
  ⇒ keine Verwirrung mit alten Infos und erhöhte Informationssicherheit
- Empfangslaufnummer: Um 1 höher ⇒ nächstes Byte „anfordern“
- Beide Übertragungsrichtungen praktisch unabhängig
  ⇒ eigener Nummernraum, Aufbau, Abbau...

Bestätigungsmechanismus



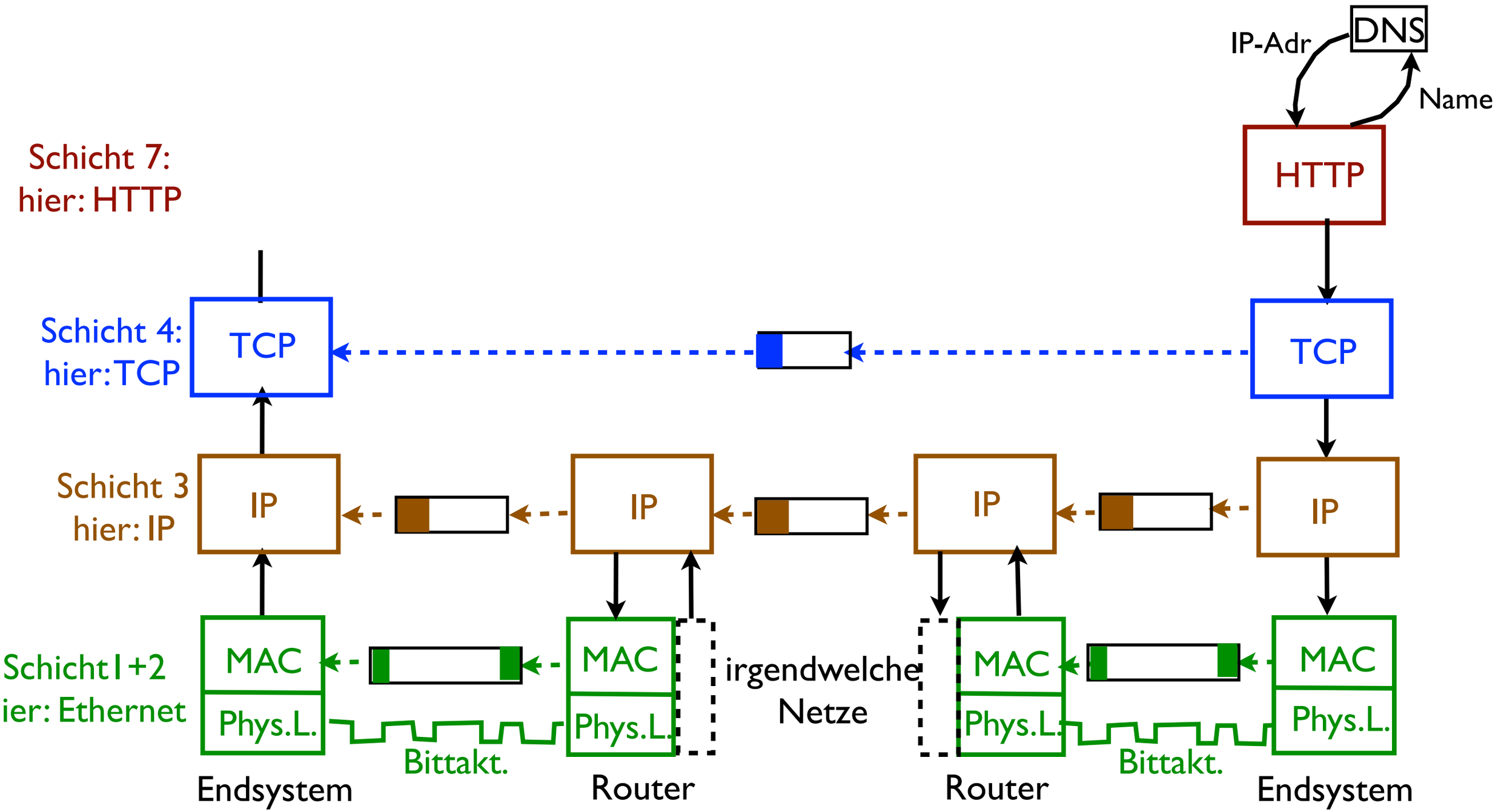
- Flusskontrolle von Bestätigungsmechanismus entkoppelt
  ⇒ keine Verzögerung von Bestätigungen sinnvoll
- Window relativ zur Empfangslaufnummer angeben
  ⇒ bestimmt gültige Sendelaufnummern (sliding window)

TCP Congestion Control (Staukontrolle)



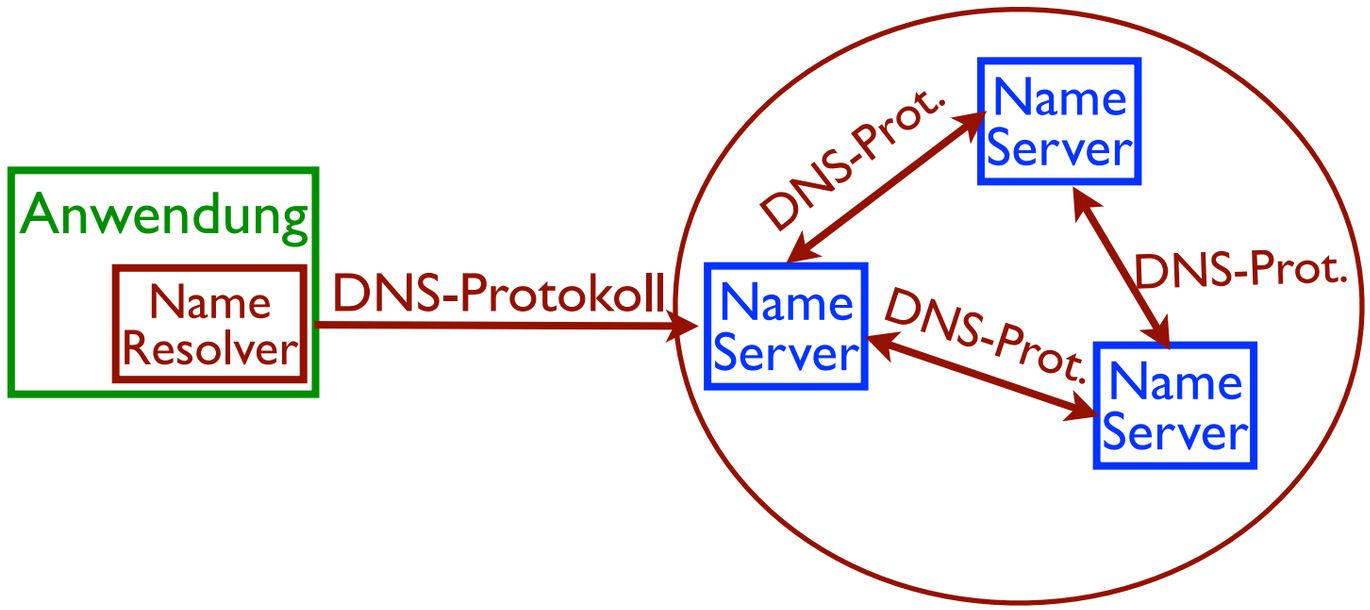
Vereinfachter Kommunikationsablauf einer Web-Nachricht

Server ← Web-Nachricht ← Client



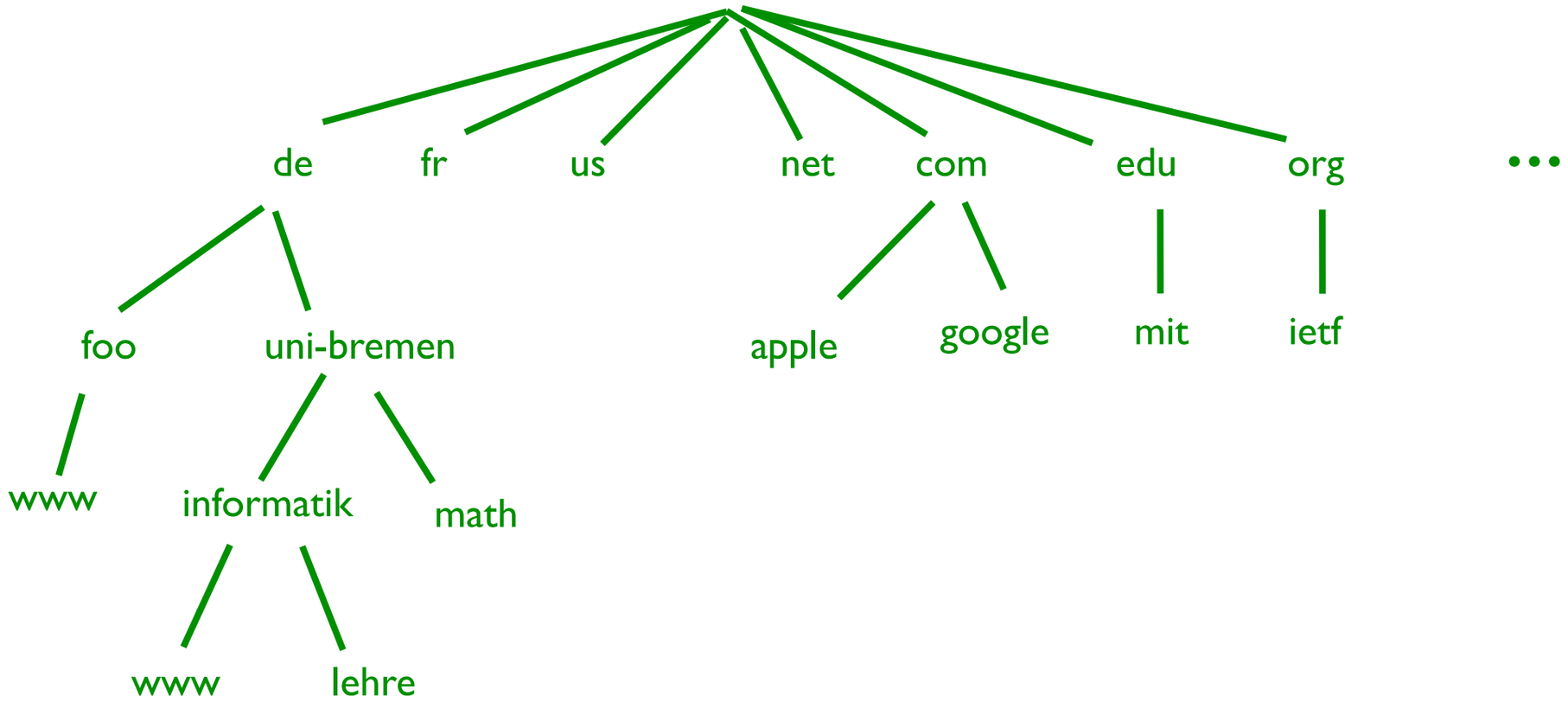
Domain Name System (DNS)

- In den Anwendungen:
  Verwendung von Domain Names statt Internet-Adressen
- Für IP Umsetzung nötig



- Verteilter Datenbestand
- Hierarchisch aufgebaut
- Für Internet-Zwecke ausgelegt, aber auch allgemeiner einsetzbar
- Durch Name-Server verwaltet (Client-/Server-Modell)
- Verweise zwischen Name-Servern
- Chaining, Referral
- Oberste Ebene des Namensraums vorgegeben (Top Level Domains), darunter von jeweiliger Organisation bestimmbar

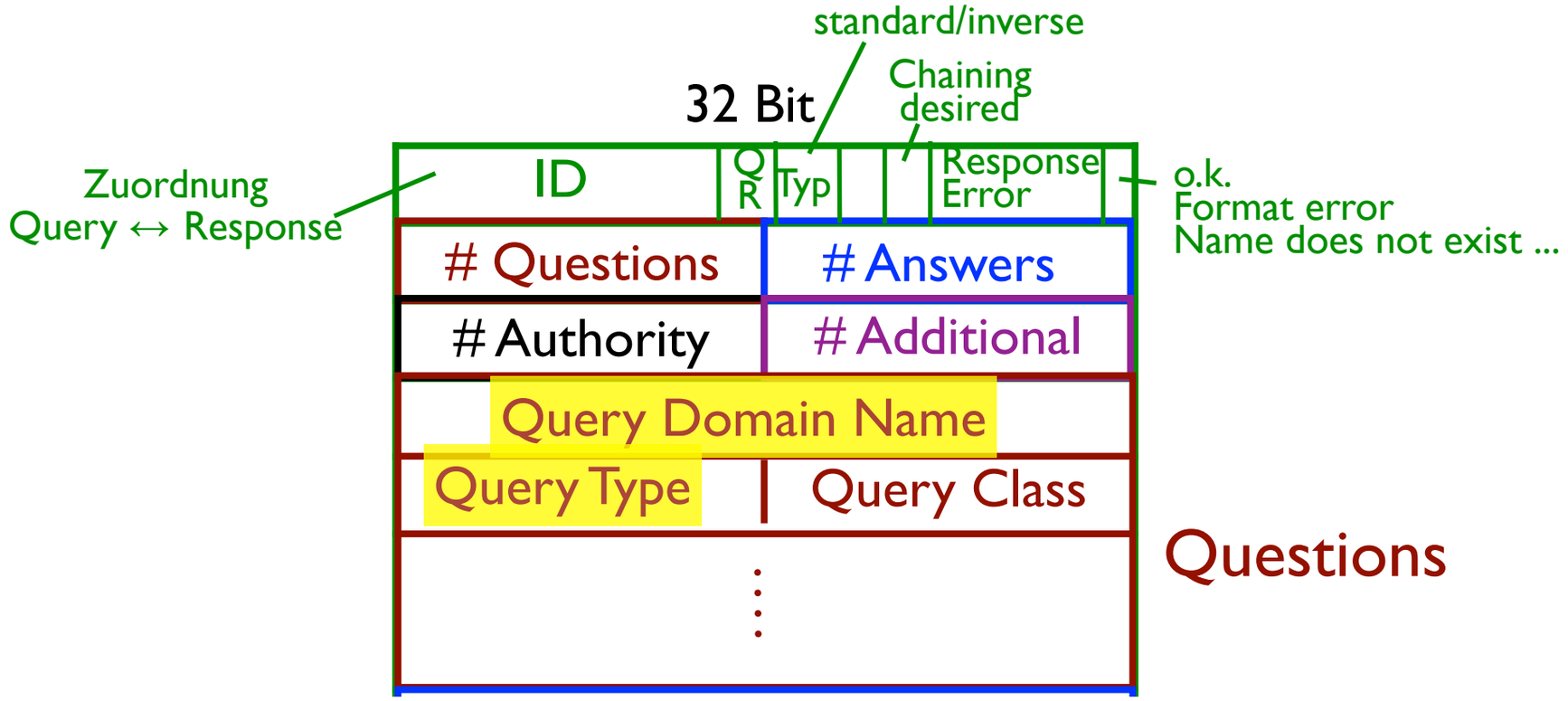
Domain Name System: Namensraum



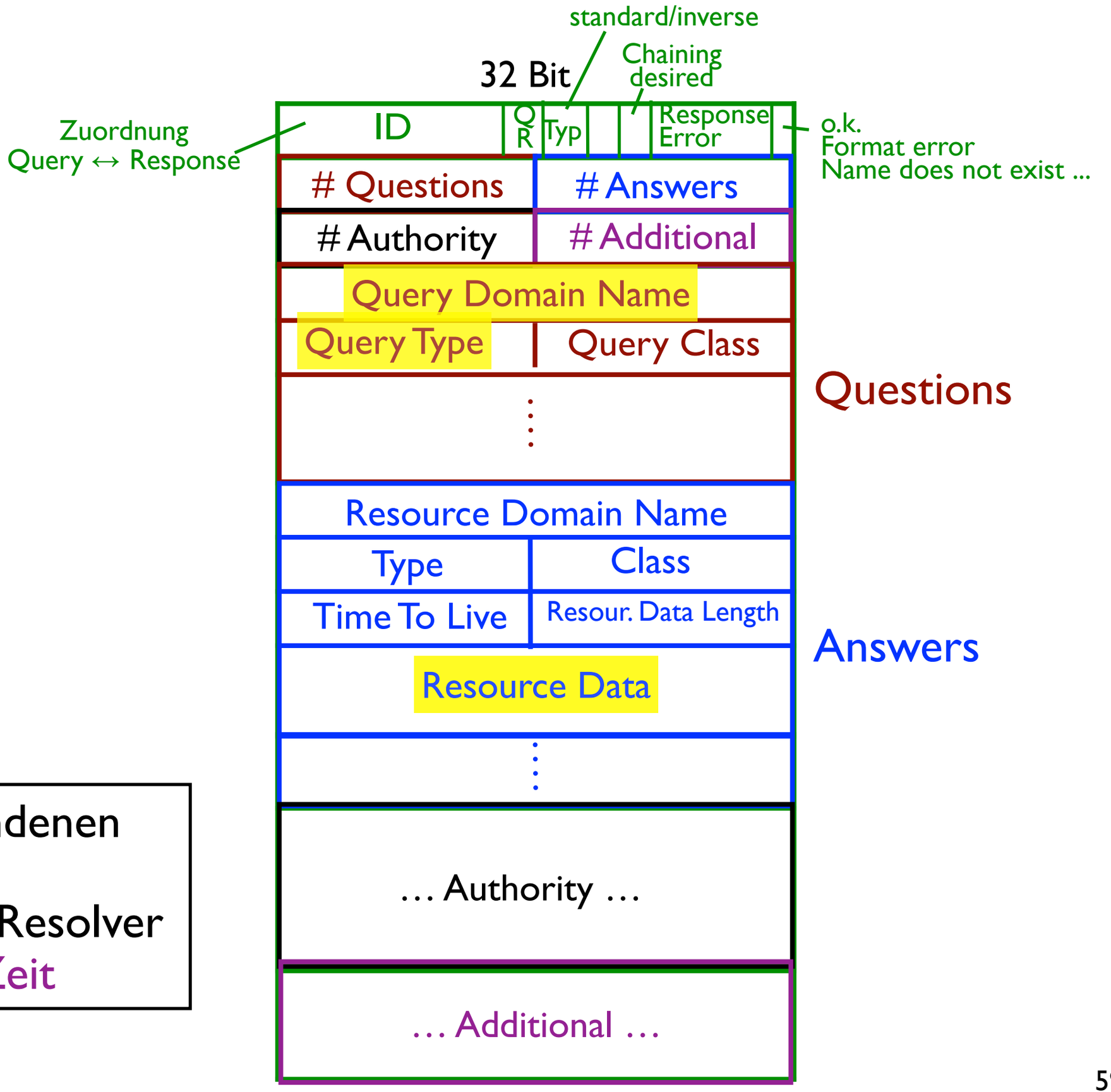
Beispiel: www.foo.de

+ diverse neuere Top Level Domains...

Nachrichtenformat des DNS-Protokolls



Nachrichtenformat des DNS-Protokolls



Caching von aufgefundenen Informationen im Name Server/Name Resolver
⇒ veralten mit der Zeit

DNS Record Types

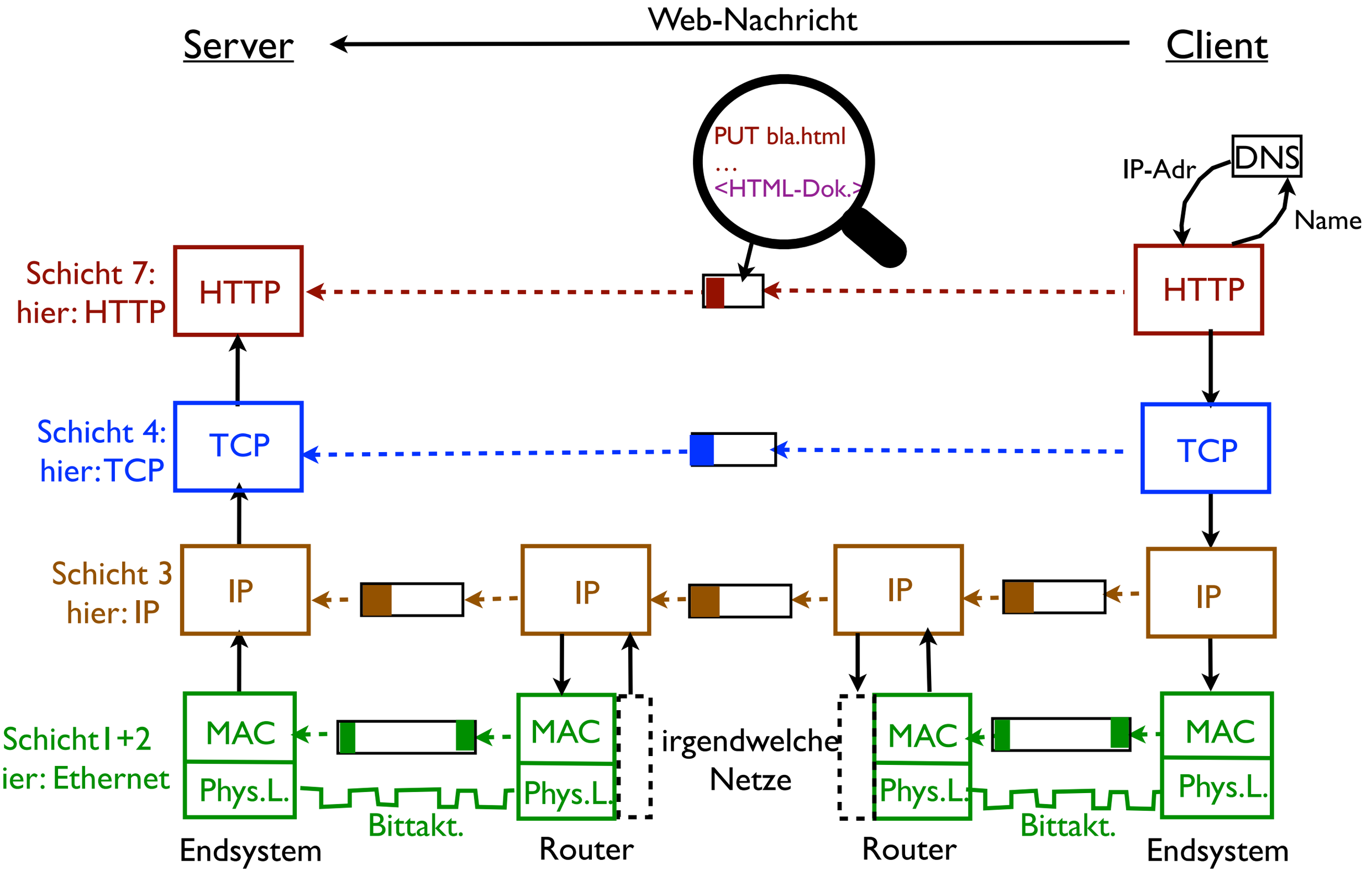
A	IPv4-Adresse (Domain Name → IPv4)
AAAA	IPv6-Adresse (Domain Name → IPv6)
PTR	inverse Anfrage (IPv4/IPv6 → Domain Name)
CNAME	Alias
TXT	Text
MX	Mail Exchanger
SRV	(allgemein: Verweis auf bestimmten Dienst, z.B. Drucker)
…	…

Fragen – Teil 3

- Welche wesentlichen Aufgaben hat TCP?
- Was kann man mit Hilfe von DNS ermitteln?

Teil 4:
Web-Anwendung

Vereinfachter Kommunikationsablauf einer Web-Nachricht



WWW (World Wide Web ➔ „Web“)

- Informationen (Dokumente) weltweit verfügbar machen
  - Internet als weltweite Infrastruktur
  - klassisch: Zugriffsprotokoll wie FTP + PostScript-Informationen

- Im Web stattdessen:
  - bearbeitbare Dokumente
  - Verweise zwischen Dokumenten

WWW (World Wide Web ➔ „Web“)

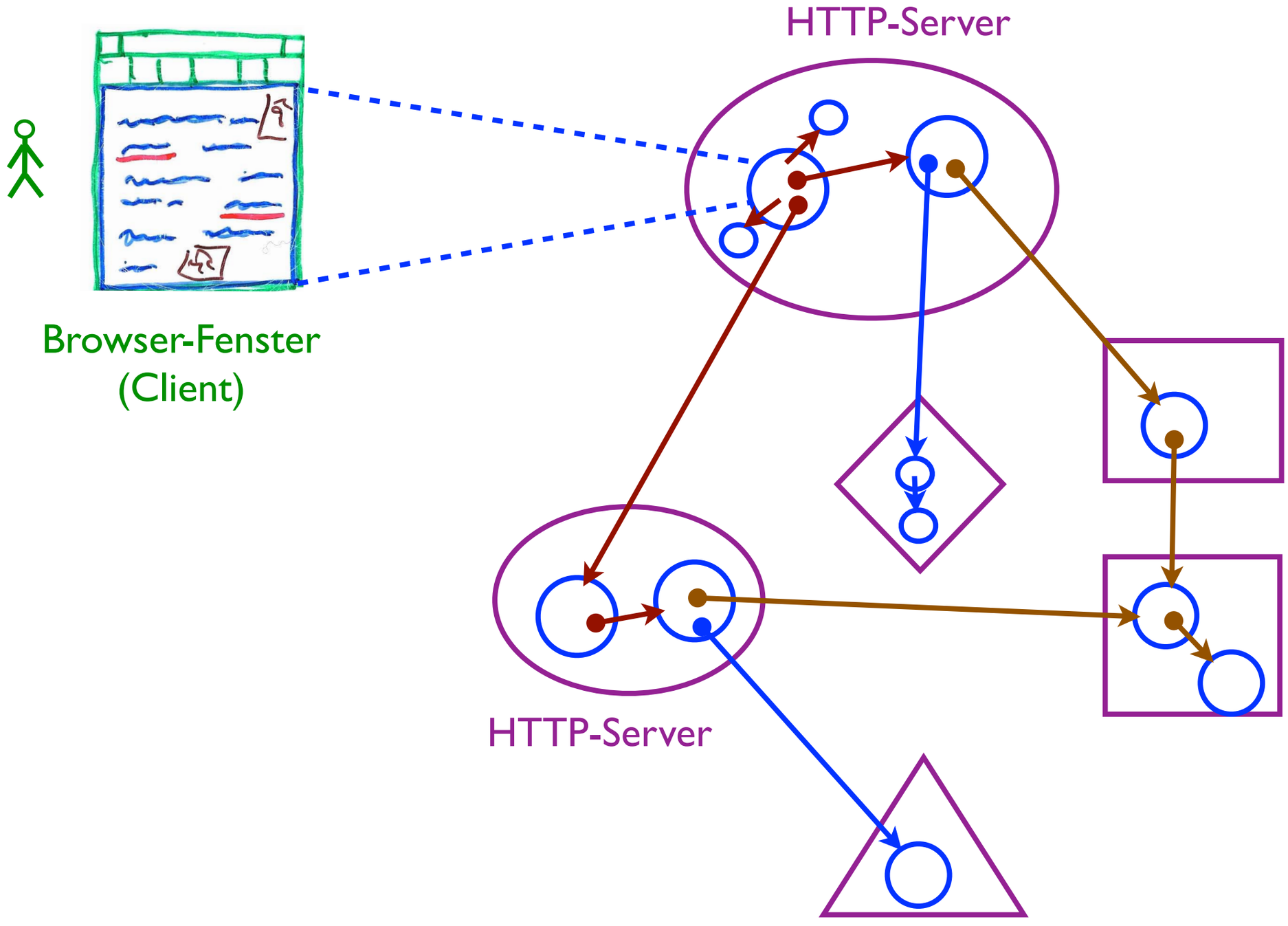
- Informationen (Dokumente) weltweit verfügbar machen
  - Internet als weltweite Infrastruktur
  - klassisch: Zugriffsprotokoll wie FTP + PostScript-Informationen

- Im Web stattdessen:
  - bearbeitbare Dokumente
  - Verweise zwischen Dokumenten

- Bausteine:
  a) geeignetes Dokumentformat
     HTML: Hypertext Markup Language
  b) eindeutige Identifikation der Dokumente
     URL: Uniform Resource Locator ⇒ URI: Uniform Resource Identifier
  c) geeignetes Protokoll
     HTTP: Hypertext Transfer Protocol

- Tim Berners-Lee, CERN

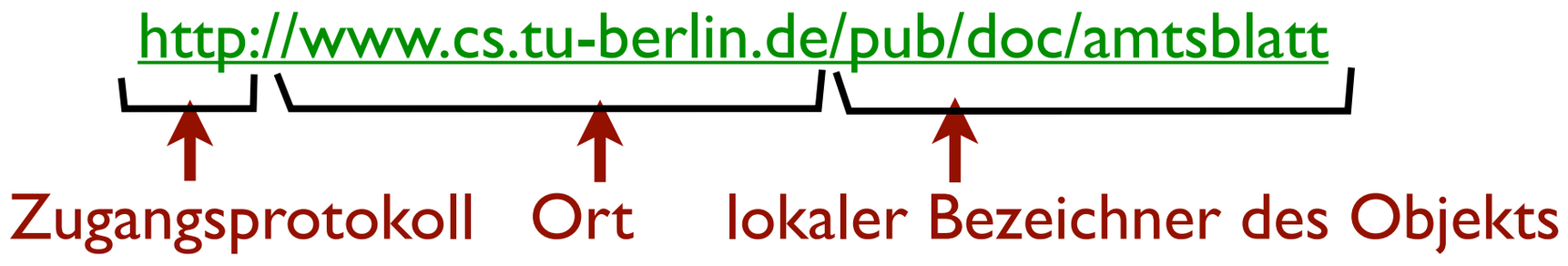
Basiert auf Client-/Server-Modell
Grundidee:



- auch andere Formate einbettbar...

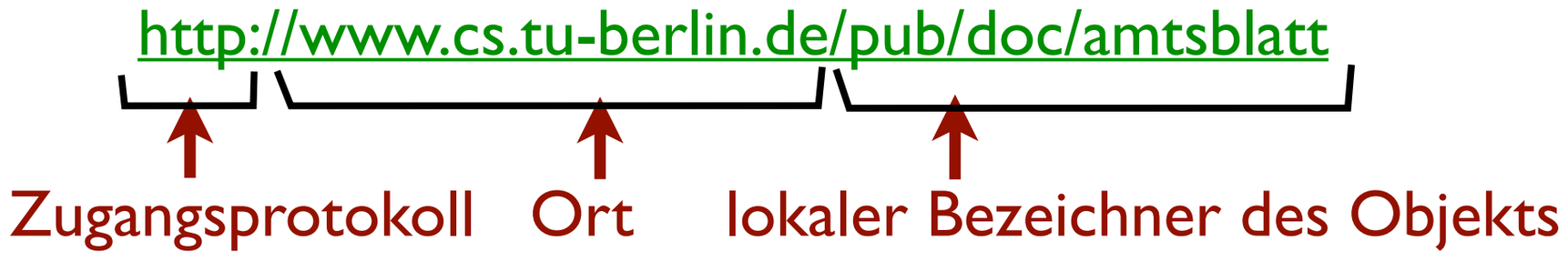
URL (Uniform Resource Locator) (⇒ allg: URI)

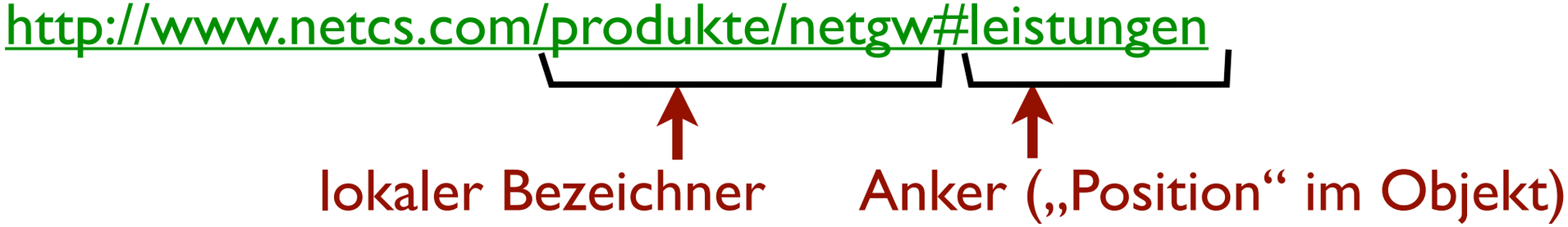
- Eindeutige Bezeichnung von Web-Objekten
- Enthalten auch Ortsangabe (Locator)



URL (Uniform Resource Locator) (⇒ allg: URI)

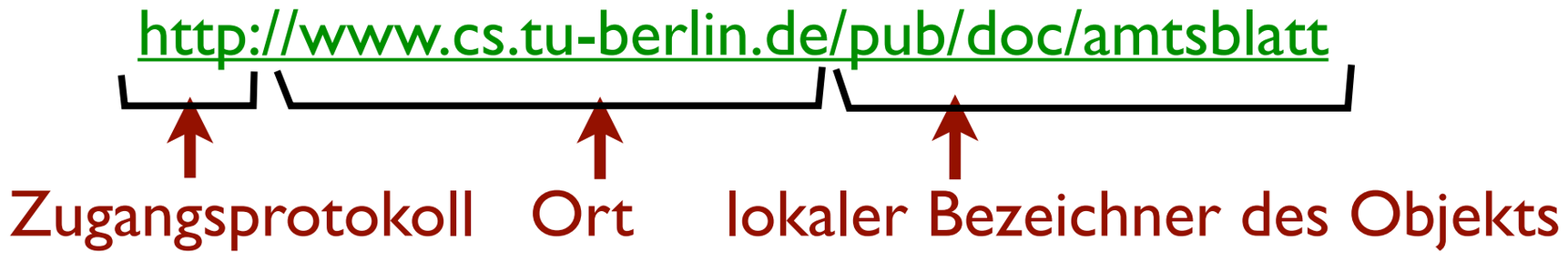
- Eindeutige Bezeichnung von Web-Objekten
- Enthalten auch Ortsangabe (Locator)

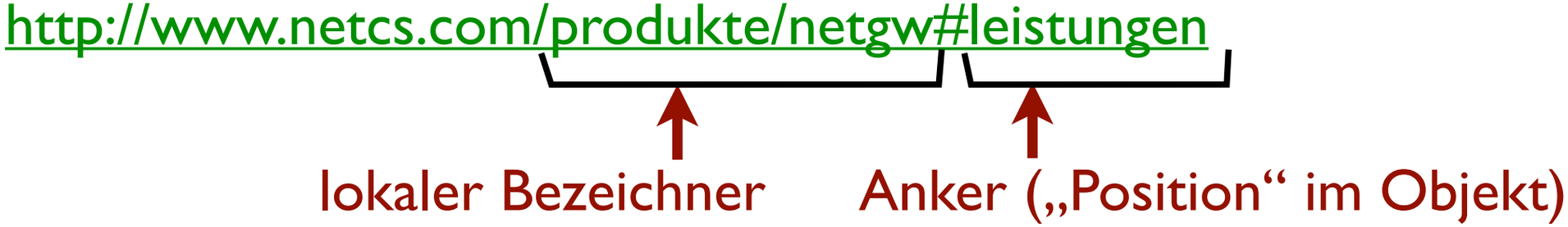




URL (Uniform Resource Locator) (⇒ allg: URI)

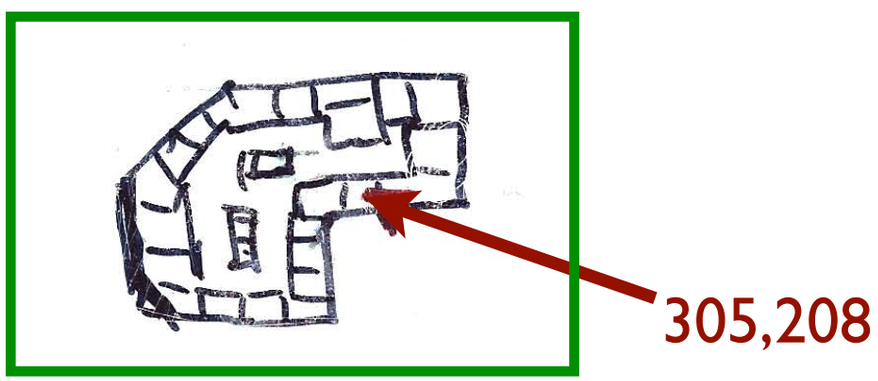
- Eindeutige Bezeichnung von Web-Objekten
- Enthalten auch Ortsangabe (Locator)





- Anfrageprotokoll:

.../fb3/raeume/ebene5?x=305,y=208
⇒ z.B. Seite über MZH 5190



.../suub/buecher/informatik?tanenbaum
⇒ z.B. Indexseite mit Büchern von A.S. Tanenbaum

HTML (Hypertext Markup Language)

- Dokument mit Auszeichnungsinformationen versehen ⇒ Dokumentstruktur
- Basiert auf spezieller SGML-DTD (Dokumenttyp-Spezifikation)
- Mittlerweile diverse Weiterentwicklungen ⇒ ... HTML5

HTML (Hypertext Markup Language)

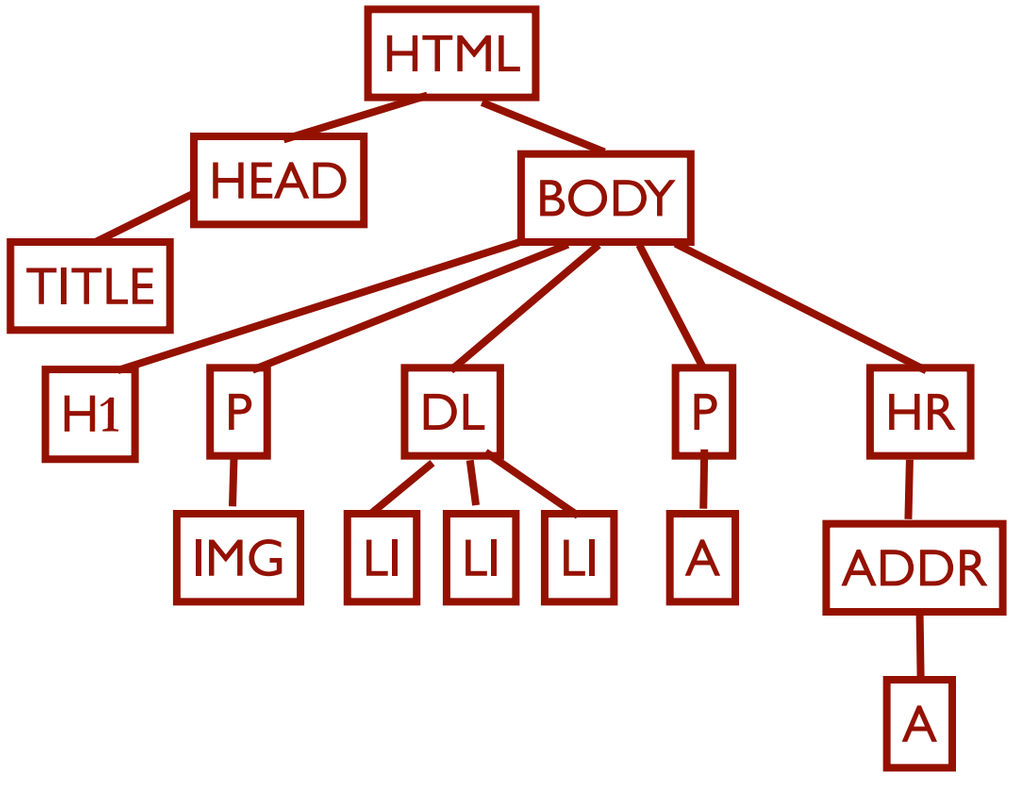
- Dokument mit Auszeichnungsinformationen versehen ⇒ Dokumentstruktur
- Basiert auf spezieller SGML-DTD (Dokumenttyp-Spezifikation)
- Mittlerweile diverse Weiterentwicklungen ⇒ ... HTML5
- HTML1-Beispiel:

```
<HTML>
<HEAD><TITLE>Uni Bremen Pterodactyli Home Page</TITLE></HEAD>
<BODY>
<H1>Care and Feeding of Pterodactyli</H1>
<P>Pterodactyli <IMG SRC="http://ucsd.edu/ptero.gif"> are nice pets:</P>
<UL>
<LI>They don't make noises.</LI>
<LI>They don't each very much.</LI>
<LI>They don't need much space.</LI>
</UL>
<P>This ist because they became extinct in
<A HREF="/geology/jura.html">jurassic</A> times.</P>
<HR>
<ADDRESS>
<A HREF="/people/cabo.html">Carsten Bormann</A>
</ADDRESS></HR>
</BODY></HTML>
```

HTML (Hypertext Markup Language)

- Dokument mit Auszeichnungsinformationen versehen ⇒ Dokumentstruktur
- Basiert auf spezieller SGML-DTD (Dokumenttyp-Spezifikation)
- Mittlerweile diverse Weiterentwicklungen ⇒ ... HTML5
- HTML1-Beispiel:

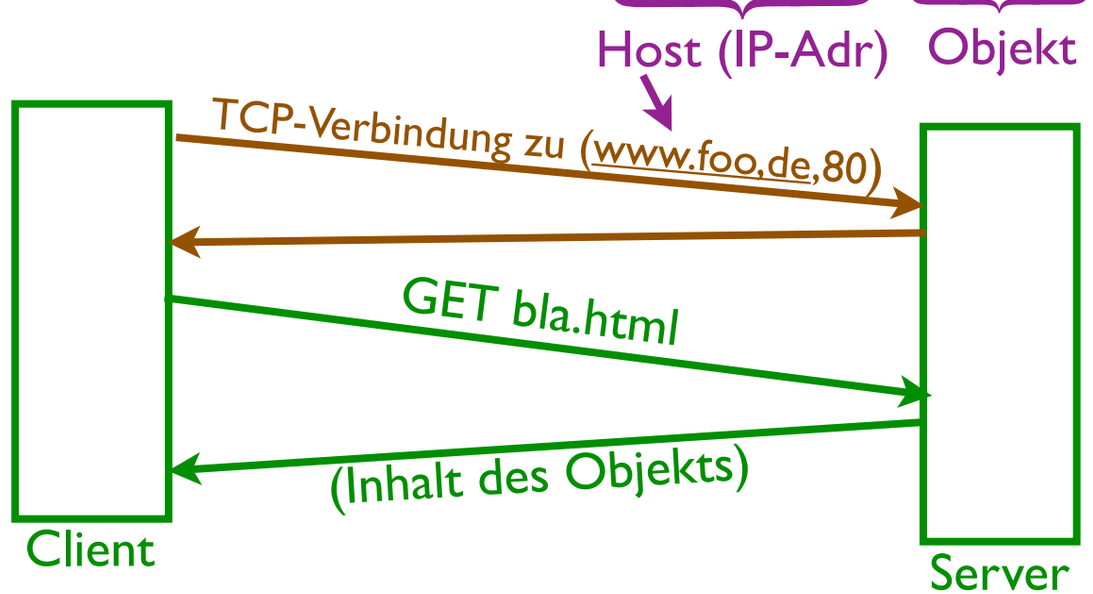
```
<HTML>
<HEAD><TITLE>Uni Bremen Pterodactyli Home Page</TITLE></HEAD>
<BODY>
<H1>Care and Feeding of Pterodactyli</H1>
<P>Pterodactyli <IMG SRC="http://ucsd.edu/ptero.gif"> are nice pets:</P>
<UL>
<LI>They don't make noises.</LI>
<LI>They don't each very much.</LI>
<LI>They don't need much space.</LI>
</UL>
<P>This ist because they became extinct in
<A HREF="/geology/jura.html">jurassic</A> times.</P>
<HR>
<ADDRESS>
<A HREF="/people/cabo.html">Carsten Bormann</A>
</ADDRESS></HR>
</BODY></HTML>
```



HTTP (Hypertext Transfer Protocol)

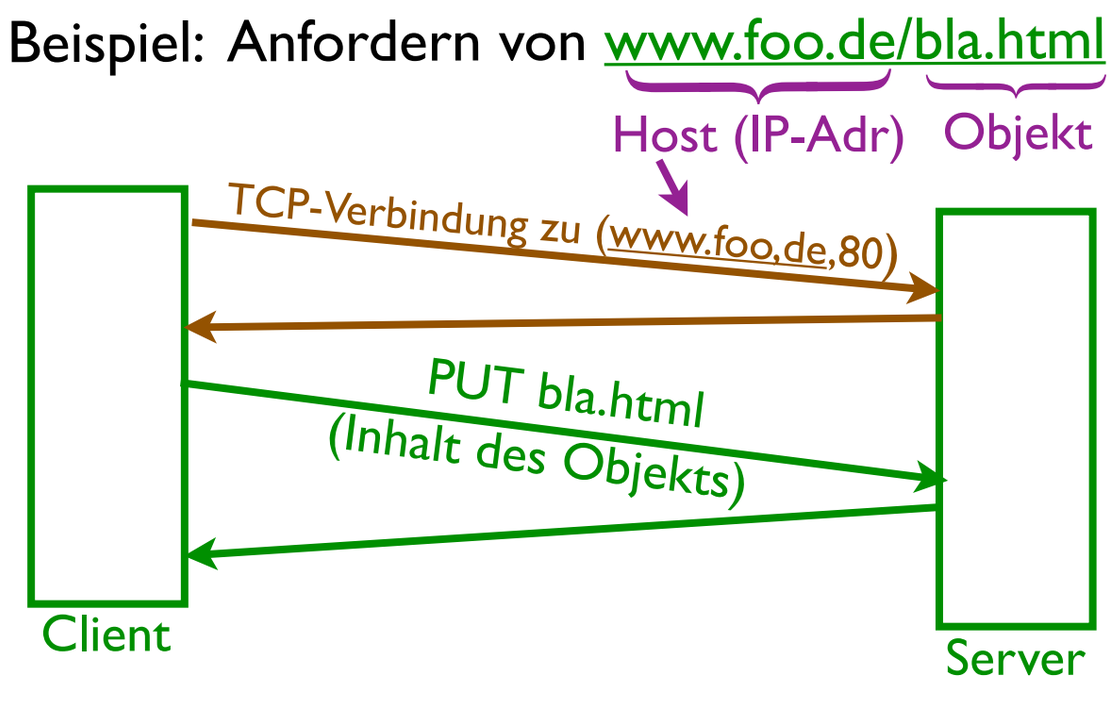
- Zugangsprotokoll zu Web-Informationen
- Klassische Version 0.9:
  Sehr einfache Grundarbeitsweise

Beispiel: Anfordern von www.foo.de/bla.html



HTTP (Hypertext Transfer Protocol)

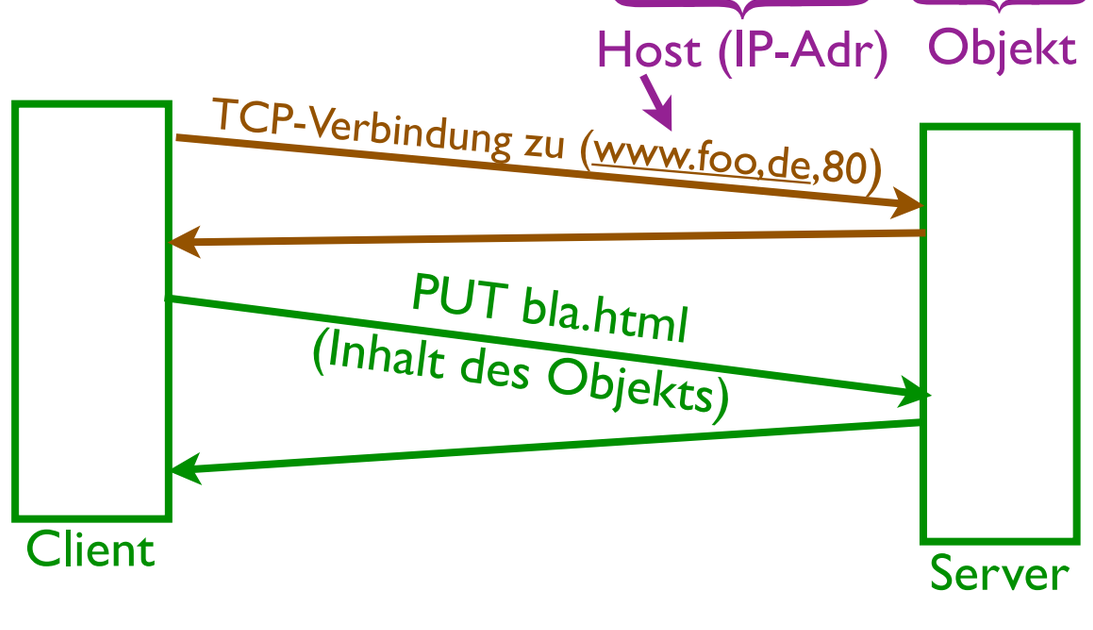
- Zugangsprotokoll zu Web-Informationen
- Klassische Version 0.9:
  Sehr einfache Grundarbeitsweise
- Auch andere Datenformate
  möglich (z.B. GIF)
- Analog für das Ablegen eines
  Objekts auf Server: PUT
- Analog für das Übertragen eines
  ausgefüllten Web-Formulars: POST



HTTP (Hypertext Transfer Protocol)

- Zugangsprotokoll zu Web-Informationen
- Klassische Version 0.9:
  Sehr einfache Grundarbeitsweise
- Auch andere Datenformate möglich (z.B. GIF)
- Analog für das Ablegen eines Objekts auf Server: PUT
- Analog für das Übertragen eines ausgefüllten Web-Formulars: POST

Beispiel: Anfordern von www.foo.de/bla.html



Probleme des „klassischen“ HTTP:

- Pro IP-Adresse nur ein Web-Server möglich
- Pro TCP-Verbindung nur ein Objekt anforderbar (eine Seite mit 3 Bildern
  → 4 TCP-Verbindungen, Bilder erst anforderbar, wenn Seite da)

  ⇒ dauert lange (TCP slow start), überlastet Netze
- Keine Aushandlung von Objekt-Formaten/-Kodierungen

HTTP (genauer)

- Client-/Server-Modell
- klassisch: ASCII-basiert
- n-buchstabige Kommandos (+ Parameter):

  GET
  PUT
  POST
  DELETE ...

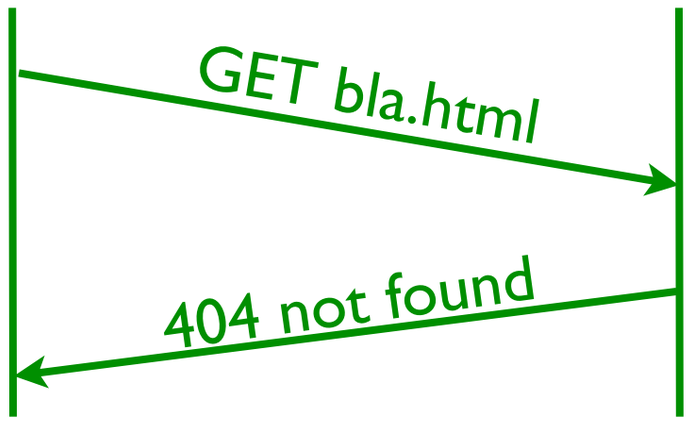
HTTP (genauer)

- Client-/Server-Modell
- klassisch: ASCII-basiert
- n-buchstabige Kommandos (+ Parameter):

  GET
  PUT
  POST
  DELETE ...

- 3-ziffrige Antwortcodes (+ Kommentare)

  1*xx* noch in Arbeit
  2*xx* o.k.
  3*xx* sinnvolle Anfrage, aber keine Antwort
  4*xx* Client-Fehler
  5*xx* Server-Fehler



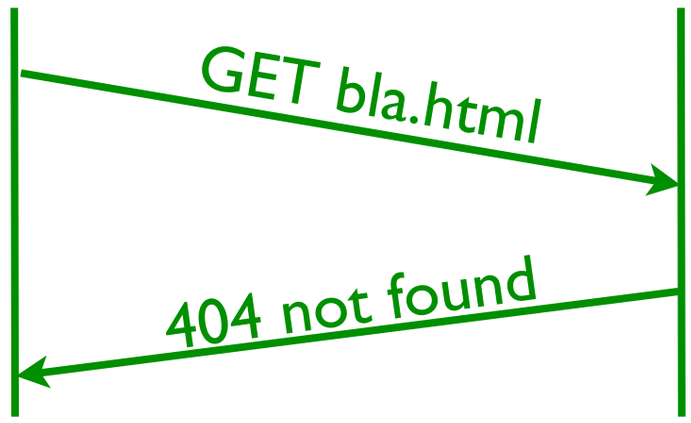
HTTP (genauer)

- Client-/Server-Modell
- klassisch: ASCII-basiert
- n-buchstabige Kommandos (+ Parameter):

  GET
  PUT
  POST
  DELETE ...

- 3-ziffrige Antwortcodes (+ Kommentare)

  1*xx* noch in Arbeit
  2*xx* o.k.
  3*xx* sinnvolle Anfrage, aber keine Antwort
  4*xx* Client-Fehler
  5*xx* Server-Fehler



- + weitere „Headerzeilen“ bei Anfrage/Antwort, die Anfrage/Ressource/... weiter beschreiben, z.B. Content-Type, Content-Language, ...
- ggf. Inhalt der Ressource (Body)

HTTP (genauer)

- Client-/Server-Modell
- klassisch: ASCII-basiert
- n-buchstabige Kommandos (+ Parameter):

  GET
  PUT
  POST
  DELETE ...

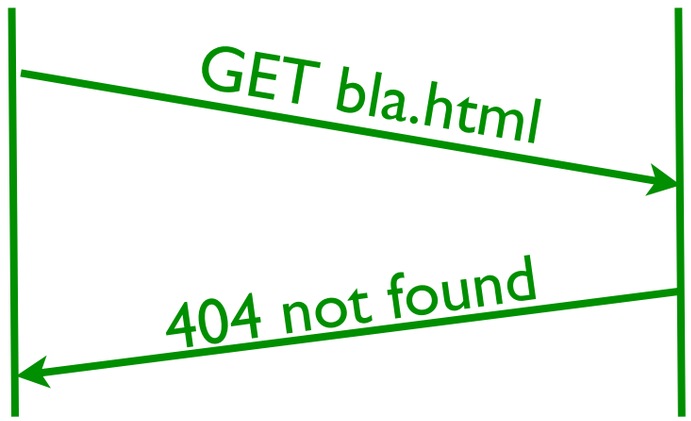
REST (vereinfacht):

HTTP-Kommandos statt anwendungsspezifische RPCs verwenden

(z.B.: Zugriff auf entferntes Dateisystem)

- 3-ziffrige Antwortcodes (+ Kommentare)

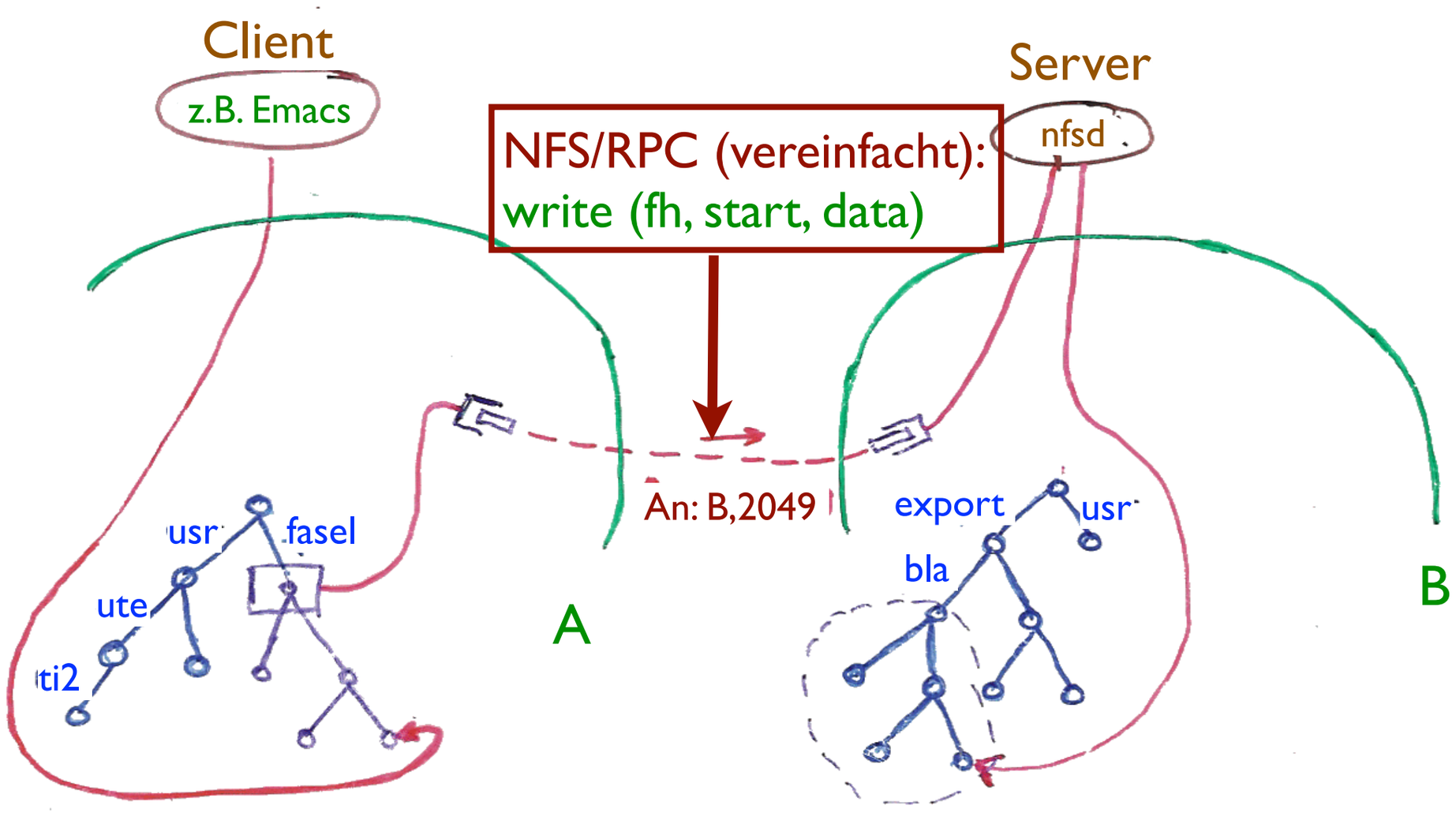
  1*xx* noch in Arbeit
  2*xx* o.k.
  3*xx* sinnvolle Anfrage, aber keine Antwort
  4*xx* Client-Fehler
  5*xx* Server-Fehler



- + weitere „Headerzeilen“ bei Anfrage/Antwort, die Anfrage/Ressource/... weiter beschreiben, z.B. Content-Type, Content-Language, ...
- ggf. Inhalt der Ressource (Body)

Nutzung von RPC-Verfahren

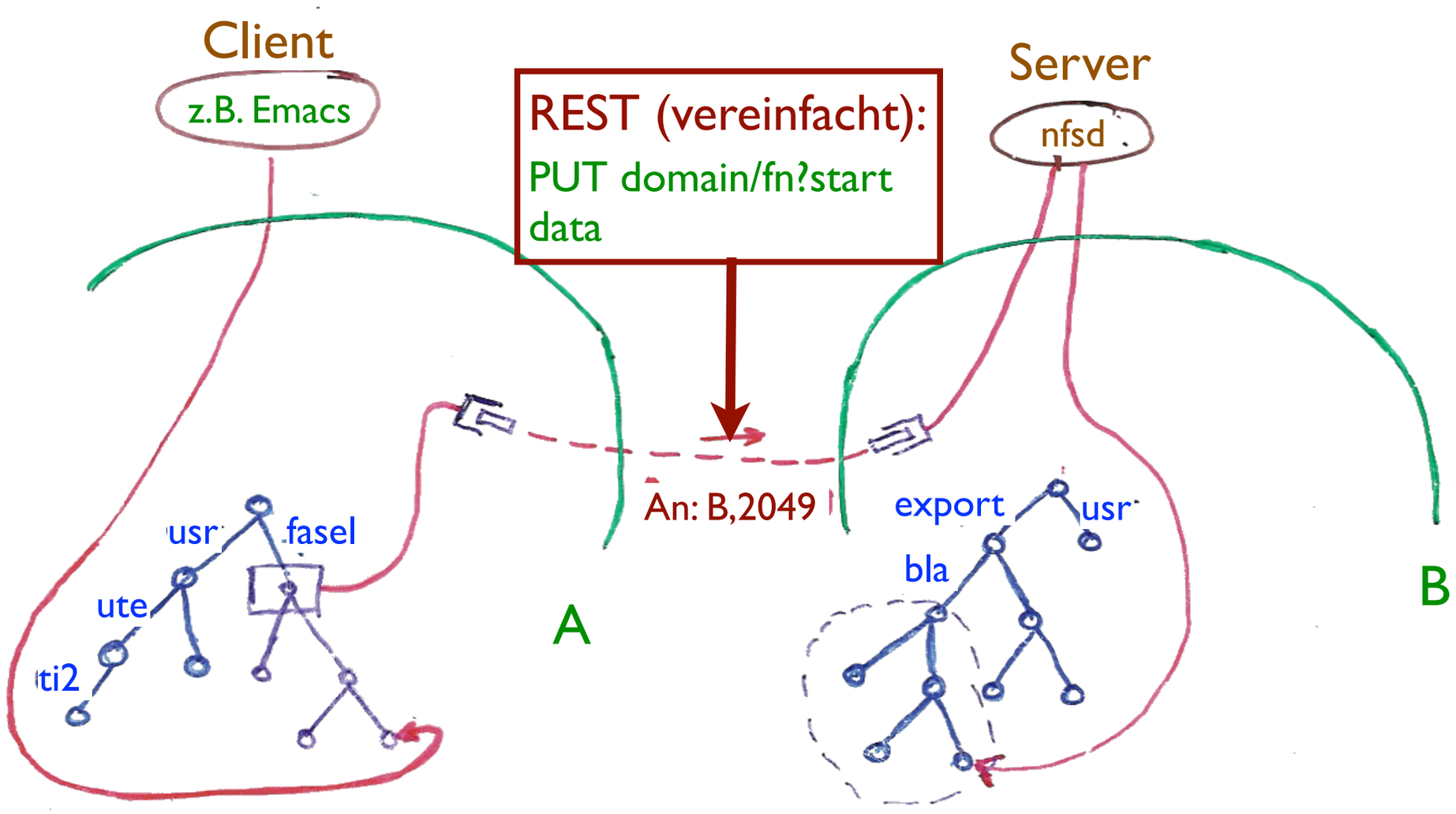
1) Zugriff auf entfernte Dateisysteme (Beispiel: NFS)



mount B:/export/bla /fasel

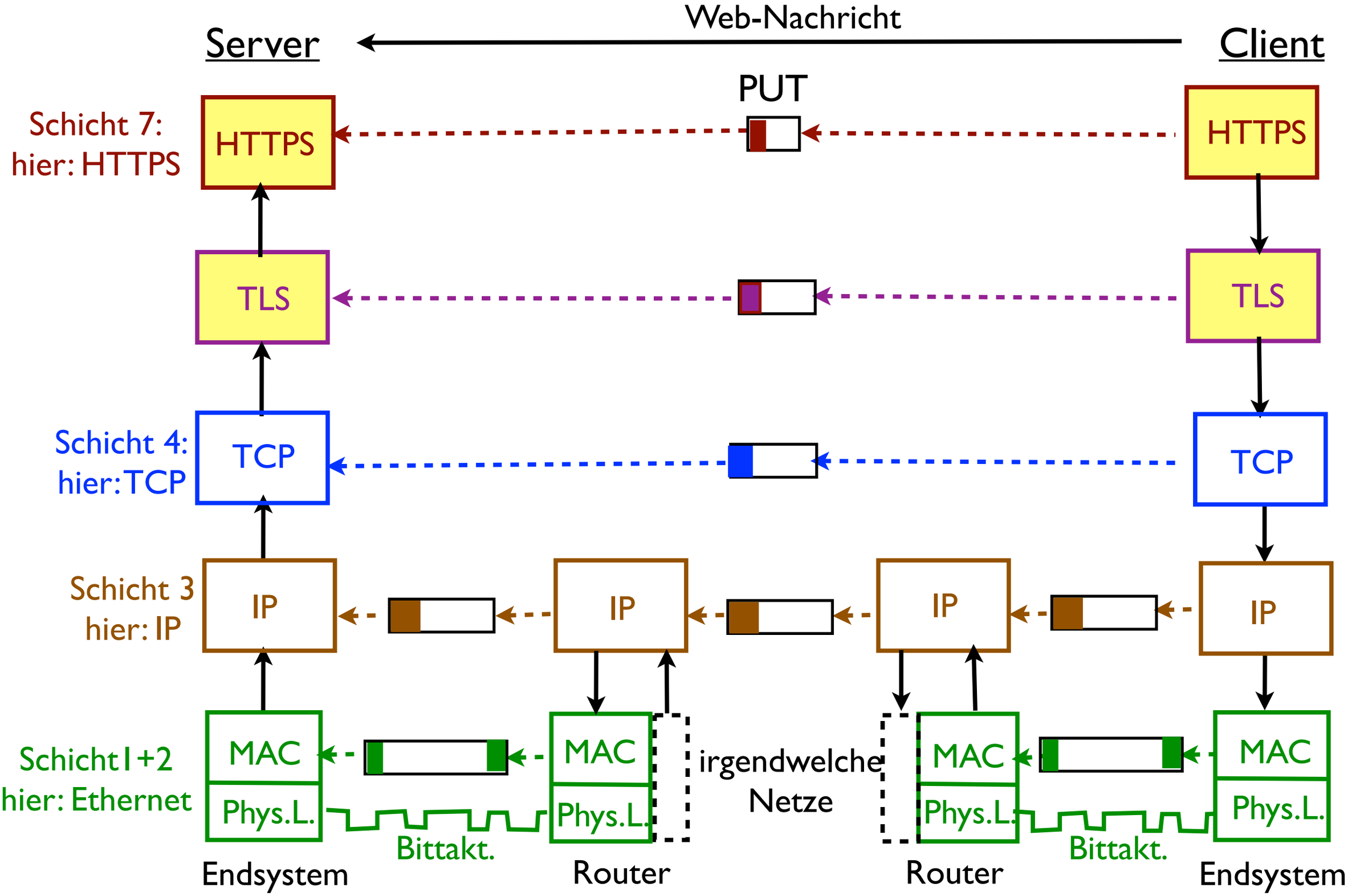
(Fiktive) Nutzung von REST

1) Zugriff auf entfernte Dateisysteme



mount B:/export/bla /fasel

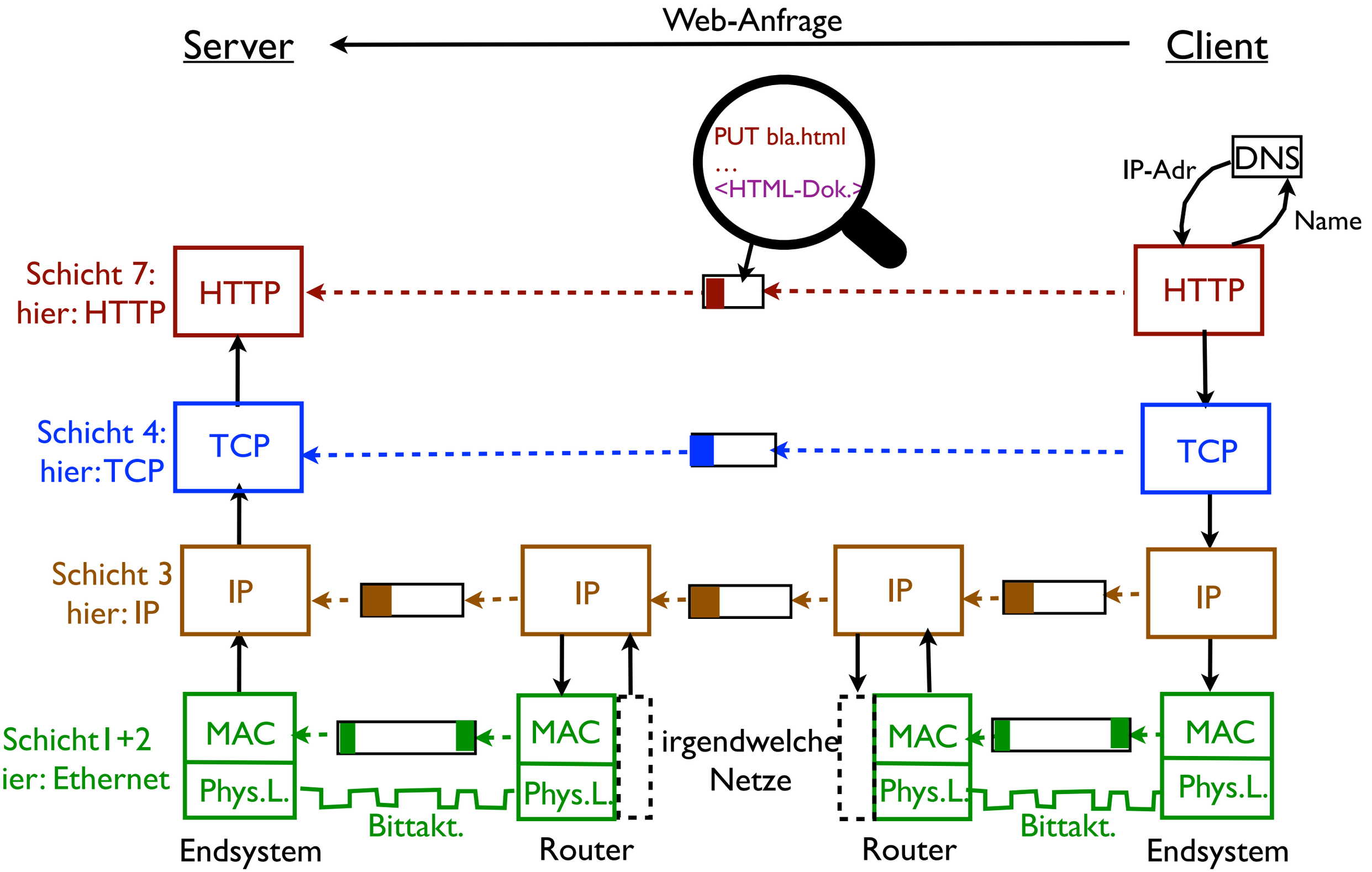
Mittlerweile meist:



Fragen – Teil 4

- Wozu benötigt man URLs?
- Welchen grundsätzlichen Aufbau hat HTTP?

Zusammenfassung



Rechnernetze 2 – Fragen

1. Wie kann eine Nachricht im Grundsatz über eine Hierarchie von Protokollen versendet werden?
2. Aus welchen Teilen besteht eine IP-Adresse?
3. Worin unterscheiden sich IPv4 und IPv6?
4. Wozu enthalten IP-Pakete ein TTL-Feld?
5. Welche wesentlichen Aufgaben hat TCP?
6. Was kann man mit Hilfe von DNS ermitteln?
7. Wozu benötigt man URLs?
8. Welchen grundsätzlichen Aufbau hat HTTP?