Rechnerkommunikation II

1. Grundmodell der Telekommunikation

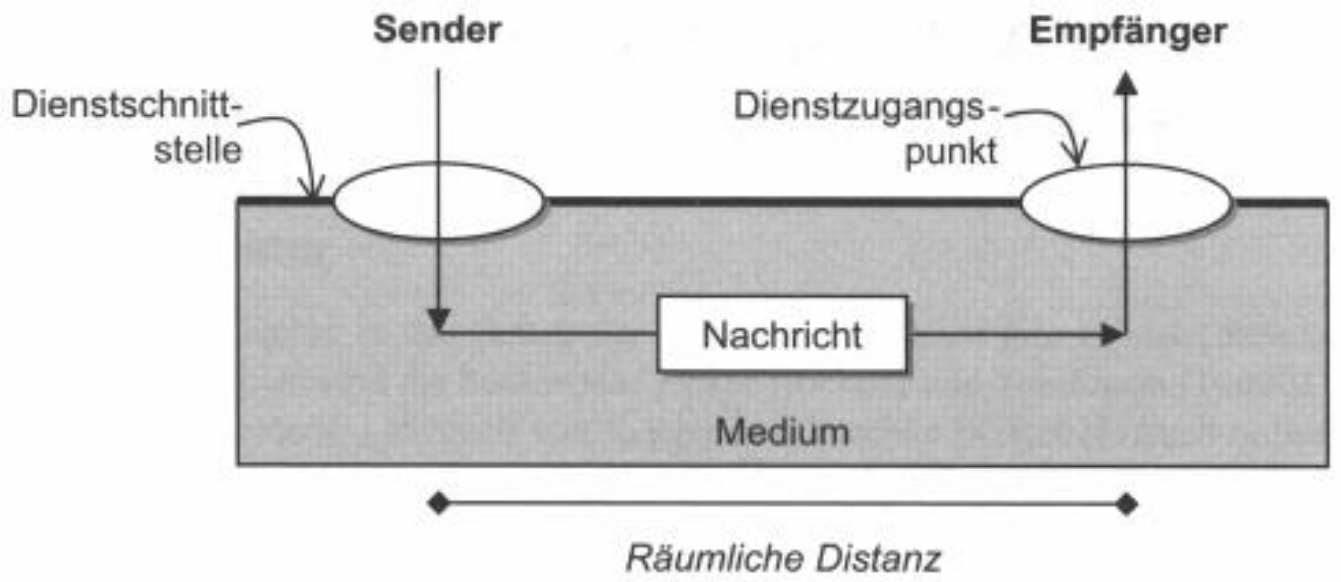


Abb. 4–2 *Grundmodell der Telekommunikation*

(Abb. aus: 1. Abeck et al.: Verteilte Informationssysteme, 2.Tanenbaum, A.: Computernetzwerke, 3. Kurose, J. u. Ross, K.:)

Rechnerkommunikation II

2. Dienst

2.1 Begriffe

- Aufgabe eines Telekommunikationssystems ist es, eine Reihe von brauchbaren, wohldefinierten und geregelten Funktionen anzubieten. Diese nennt man *Dienste*.
- Dienste werden von *Diensterbringern* angeboten.
- Der *Dienstnehmer* nimmt die Dienste in Anspruch.
- Teile eines Dienstes, *Dienstfunktionen*, können unabhängig voneinander in Anspruch genommen werden:

  z.B.:

  WWW → HTTP → Abruf einer HTML-Seite
- Einzelne Vorgänge einer Dienstfunktion werden als *Dienstprimitive* bezeichnet.
- Die Zusammensetzung einer Dienstfunktion aus Dienstprimitiven wird auch als *Dienstprozedur* bezeichnet.
- Dienste können in einer *Diensthierarchie* angeordnet werden.

Rechnerkommunikation II

2.2 Verbindungsloser und verbindungsorientierter Dienst

2.2.1 Verbindungsorientierter Dienst

- Beim Nutzen eines verbindungsorientierten Dienstes senden Client und Server Steuerpakete, *bevor* sie die echten Daten senden („Handshake“).
- Analogie: Telefonsystem

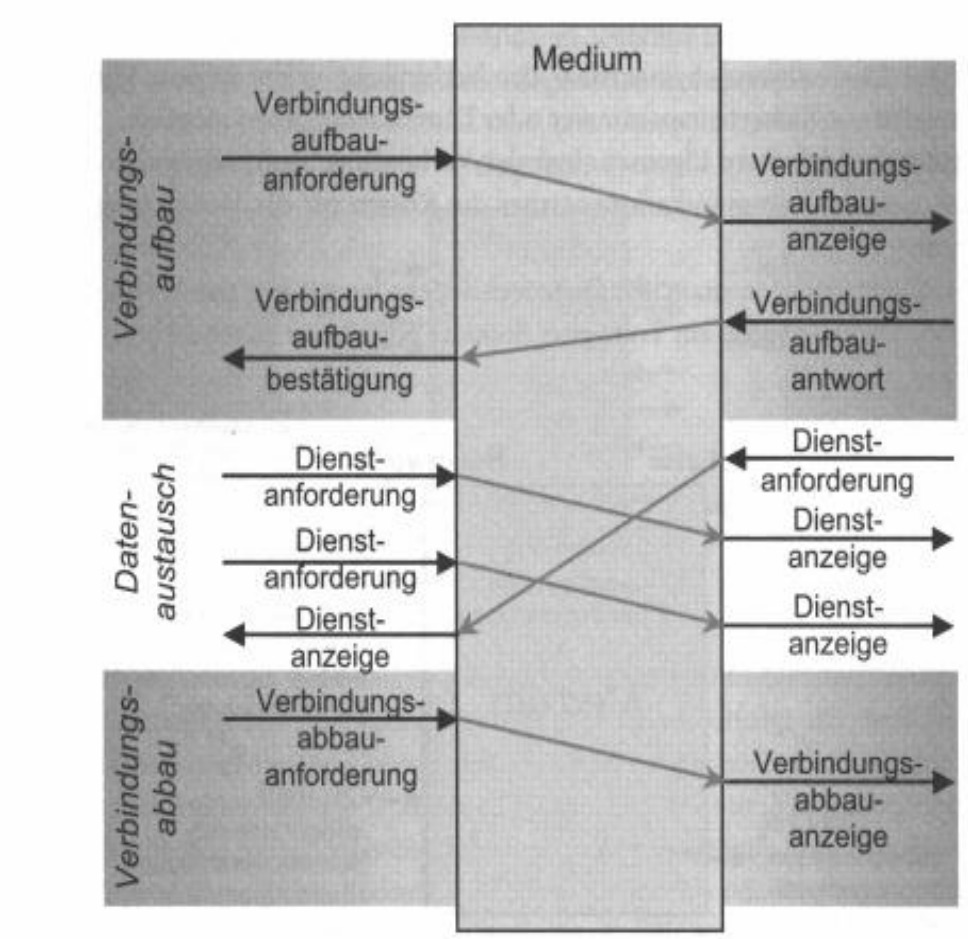


Abb. 5–5 *Phasen eines verbindungsorientierten Dienstes*

2.2.2 Verbindungsloser Dienst

- Beim verbindungslosen Dienst gibt es kein Handshake. Eine Seite schickt Pakete einfach los.
- Ohne Handshake wird die Übertragung schneller, aber es gibt auch keine Bestätigungen.
- Der Sender kann nie sicher sein, ob seine Pakete angekommen sind.
- Der Empfänger kann nie sicher sein, ob er alle Pakete fehlerfrei und in der richtigen Reihenfolge erhalten hat.
- UDP (User Datagram Protocol) stellt im Internet den verbindungslosen Dienst zur Verfügung.
- Analogie: Postsystem

2.2.3 Dienstarten

Welche Probleme könnten beim Übertragen von Daten auftauchen?

- Daten können verloren gehen
- Daten können verändert werden
- Ein Teilnehmer sendet mehr Daten bzw. schneller als der Empfänger sie verarbeiten kann.
- Das Netz kann überlastet sein, d.h. es gibt Staus.

Deshalb werden Dienste und Dienstprimitive zur Vermeidung solcher Probleme angeboten:

- Bestätigungen (Acknowledgment, ACK) und Neuübertragungen (Retransmission), z.B. bei TCP → zuverlässiger Transfer aller Daten
- Fehlerkontrolle (Fehlererkennungs-/ korrekturcodes)

Rechnerkommunikation II

- Flusskontrolle
  - Keine Seite einer Verbindung soll durch zu schnelles Senden von Paketen überschwemmt werden.
  - Die Flusskontrolle zwingt den Sender, die Rate zu reduzieren, sobald die Gefahr der Überschwemmung besteht.
  - Zur Kontrolle verwenden Sender und Empfänger Sende- und Empfangspuffer.

- Überlastkontrolle
  - Auch weiterleitende Router können überschwemmt werden, wenn insgesamt zu viel Verkehr im Netz herrscht.
  - Ist ein Router überlastet, können seine Puffer überlaufen und Pakete verloren gehen.
    - Neuankommende Pakete können verworfen werden, d.h. sie gelangen nicht in den Puffer.
    - Pakete aus dem Puffer, die noch nicht sicher weitergeleitet werden konnten, können verworfen werden.

Rechnerkommunikation II

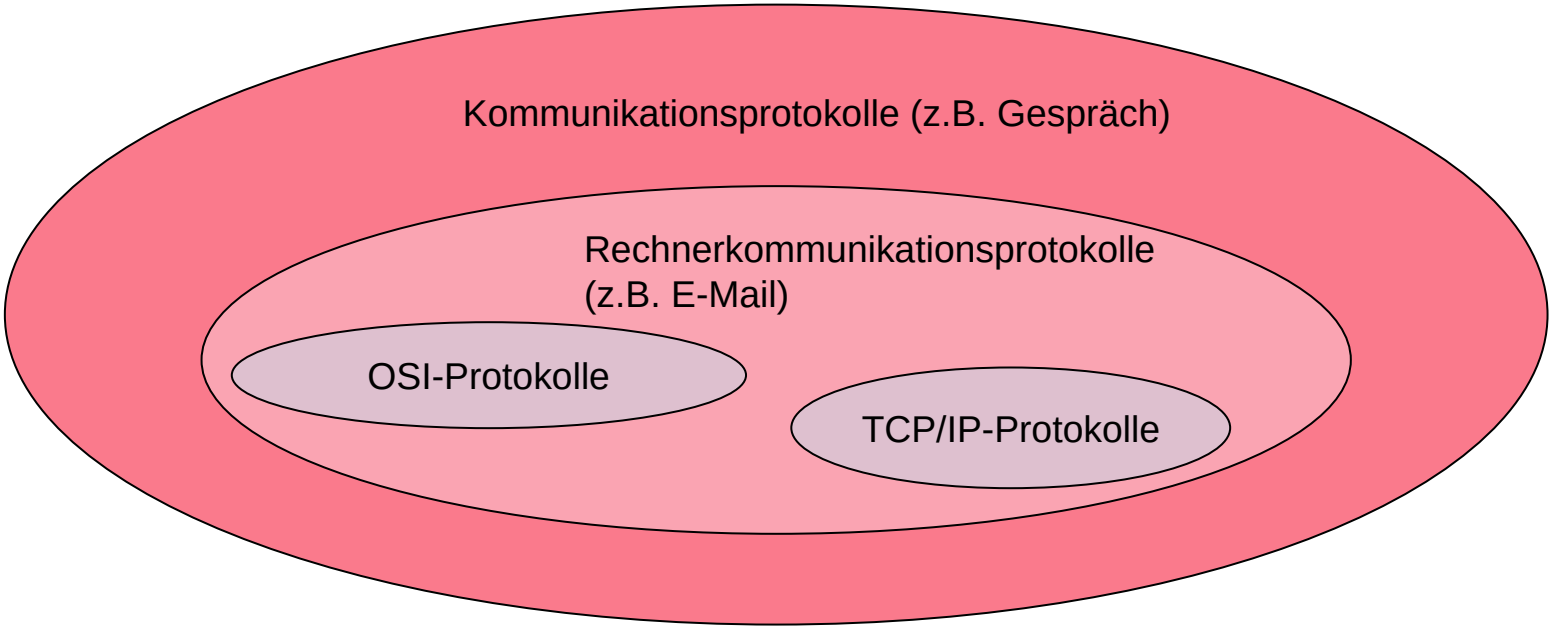
	Dienst	Beispiel
Verbindungs-orientiert	Zuverlässiger Nachrichtenstrom	Folge von Seiten
	Zuverlässiger Bytestrom	Fernanmeldung
	Unzuverlässige Verbindung	Digitalisierte Sprache
Verbindungslos	Unzuverlässiges Datagramm	Junk-E-Mail
	Bestätigtes Datagram	Registrierte E-Mail
	Anforderung/Antwort	Datenbankabfrage

Abbildung 1.16: Sechs verschiedene Dienstarten

Rechnerkommunikation II

3. Protokoll

- Damit Dienste über eine räumliche Distanz erbracht werden können, müssen Dienstnehmer und Dienstgeber (Client/ Server) nach bestimmten Regeln kommunizieren. *Protokolle* spezifizieren diese Regeln.



Rechnerkommunikation II

4. Verteiltes geschichtetes Telekommunikationssystem

- Ein Telekommunikationssystem besteht aus räumlich verteilten Teilsystemen. Diese Teilsysteme sind in *Schichten* aufgeteilt.
- Eine *Instanz* (Einheit) einer Schicht erbringt die Aufgaben einer Schicht. Die Protokollinstanzen einer Schicht n stellen der nächst höheren Schicht n+1 die Dienste zur Verfügung.
- Unter *vertikaler* Kommunikation versteht man die Inanspruchnahme von Diensten innerhalb eines Teilsystems.
- Der Austausch von Daten zweier Instanzen verschiedener Teilsysteme, die aber derselben Schicht zugehören, nennt man *horizontale* Kommunikation.
- Die Komplexität des Systems wird dadurch verringert, dass die Instanzen von Schicht n nur die darunter liegenden der Schicht n-1 kennen.

Rechnerkommunikation II

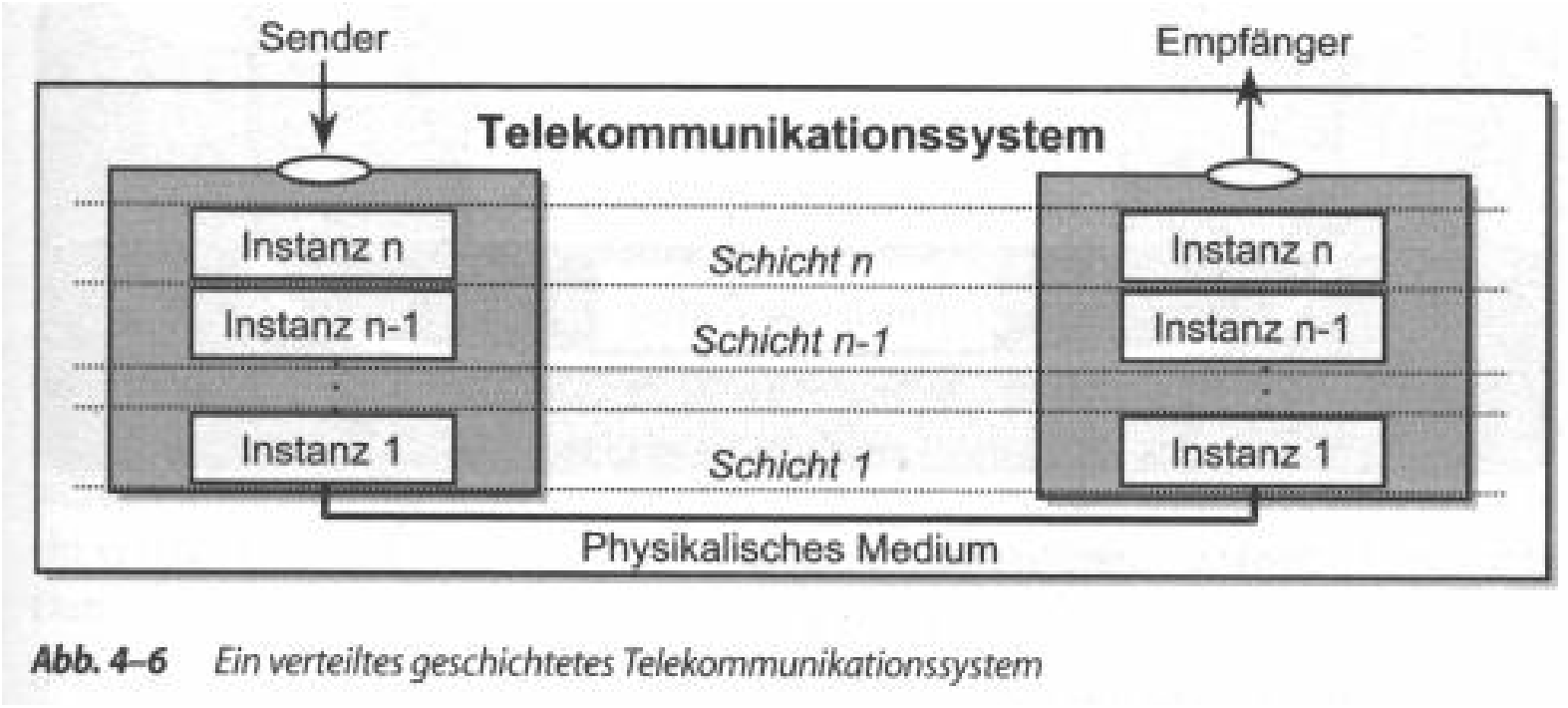


Abb. 4–6 *Ein verteiltes geschichtetes Telekommunikationssystem*

Rechnerkommunikation II

- Die Gesamtheit aller Protokolle eines Systems nennt man Protokollstapel (protocol stack).
- Protokolldateneinheiten (PDU):
  - Die Protokolle einer bestimmten Schicht *n* werden auf die Netzwerkeinheiten verteilt, d.h. in jeder Netzwerkeinheit gibt es einen Teil von Schicht *n*.
  - Diese Teile kommunizieren miteinander durch Austausch von Schicht-*n*-Nachrichten.
  - Diese Nachrichten heißen Protokolldateneinheiten (PDU, Protocol Data Units) von Schicht *n*.

Rechnerkommunikation II

5. Referenzmodelle
5.1 ISO/ OSI

7-Schichten-Modell

- oberste 3 Schichten: Anwendungssystem (Ablaufsteuerung, Informationsdarstellung)
- untere 4 Schichten: Transportsystem (Transport der Daten als Bitstrom)
- jede Schicht erfüllt genau definierte Aufgaben

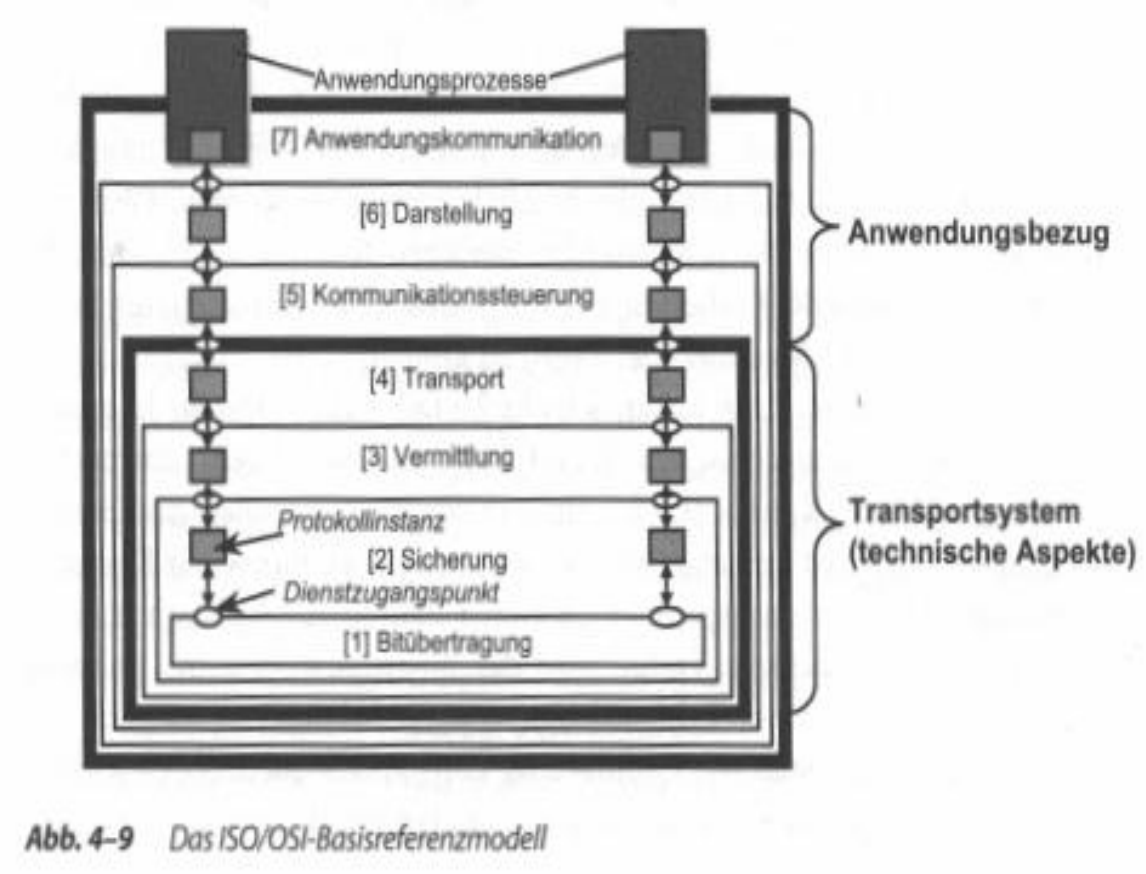


Abb. 4–9 *Das ISO/OSI-Basisreferenzmodell*

Rechnerkommunikation II

5.1.1 Die Bitübertragungsschicht (Physical Layer)

Protokolle des Physical Layers regeln die Standardisierung der mechanischen, elektrischen und Signal- Schnittstellen und somit die Übertragung von Bits über einen Kommunikationskanal.

z.B.:

- Wie viele Pins hat ein Stecker und wie ist die Belegung geregelt?
- Wie viele Volt sollen einer logische 1 entsprechen, wie viele einer 0?
- Wie viele Nanosekunden soll ein Bit dauern?
- Soll die Übertragung in beide Richtungen erfolgen?

Es existieren viele Standards; z.B.: für serielle Leitungen RS-232-C

Rechnerkommunikation II

5.1.2 Die Sicherungsschicht (Data Link Layer)

- Hauptaufgabe: Der in der physikalischen Schicht noch ungesicherte ungepufferte Übertragungskanal wird durch die Sicherungsschicht zum gesicherten Transportmedium.
- Übertragungsfehler werden erkannt und eliminiert.
- Aufteilung des Bitstroms in Datenrahmen (Frames), innerhalb derer Fehlererkennung und –korrektur möglich ist (z.B. anhand von Bitmuster, Prüfsumme, wiederholtes Senden)
- Quittieren des Frames durch Senden einer Bestätigung (Acknowledgement) des Empfängers.
- Weitere Aufgabe: Sicherung gegen Datenüberschwemmung
- Bsp. für Protokolle der Sicherungsschicht: HDLC (High-Level-Datalink-Control), PPP (Point-to-Point)

Rechnerkommunikation II

5.1.3 Die Vermittlungsschicht (Network Layer)

- Hauptaufgabe: Verbindung der in der vorhergehenden Schicht gesicherten Teilstreckenverbindungen über mehrere Knoten hinweg. Durch die Protokolle der Vermittlungsschicht ist somit ein Datentransfer über mehrere Teilnetze möglich.
- *Routing* ist das Verfahren zur Bestimmung von Streckenverbindungen (über Routing-Algorithmen). Voraussetzung: Endsysteme müssen eindeutig adressierbar sein.
- Weitere Aufgaben:
  - Vermeiden von Datenstaus.
  - Abstimmung von Adressierschema oder Paketgröße beim Übergang in ein anderes Netz.
  - Kompatibilität von Protokollen
- Bsp.: IP (Internet Protocol)

Rechnerkommunikation II

5.1.4 Die Transportschicht (Transport Layer)

- Abstrahiert von unterschiedlichen Vermittlungsschichtdiensten; Aspekte des eigentlichen Nachrichtenaustauschs bleiben somit dem Dienstbenutzer verborgen.
- Die Dienste der Transportschicht sorgen dafür, dass Daten in Datenpakete zerlegt werden bzw. die einzelnen Datenpakete wieder korrekt zusammengefügt werden, so dass sie von Anwendungen verarbeitet werden können.
- Echte *Endpunkt-zu-Endpunkt*-Schicht, weil Quell- und Zielmaschine direkt miteinander kommunizieren, während auf den niedrigeren Schichten eine Maschine immer nur mit dem unmittelbaren Nachbarn kommuniziert.
- Bsp.: TCP (Transmission Control Protocol), UDP (UserDatagram Protocol)

Rechnerkommunikation II

5.1.5 Die Sitzungsschicht (Session Layer, Kommunikationssteuerungsschicht)

- Ermöglicht die Nichtunterbrechbarkeit von Kommunikationsbeziehungen durch Sitzungen
- Dialogsteuerung (Dialogue Control) organisiert den Ablauf einer Kommunikation (wer darf zum Zeitpunkt x Daten übertragen?)
- Anwendungsbezogene Synchronisation des Informationsaustauschs über Prüfpunkte (Zurücksetzen auf diese bei anwendungsspez. Problemen)

5.1.6 Die Darstellungsschicht (Presentation Layer)

- Ermöglicht die Kommunikation zwischen Systemen mit unterschiedlicher lokaler Darstellung der Daten (z.B.: Speicherung eines int-Wertes in 16 o. 32 bit)
- Verantwortlich für die Bedeutungstreue der Kommunikationsarchitektur
- Verwaltung und Definition von abstrakten Datenstrukturen, Standardkodierungen

Rechnerkommunikation II

5.1.7 Die Anwendungsschicht (Application Layer)

- Bereitstellung der anwendungsspezifischen Dienste (z.B. Dateitransfer, E-Mail)
- Grundsätzlich sind hier alle Applikationen und Protokolle angesiedelt, die sich nicht eindeutig den beiden darunter liegenden Schichten zuordnen lassen
- Bsp.: WWW-Browser u. Web-Server → Protokoll HTTP → Dienst: Abrufen einer HTML-Seite

Rechnerkommunikation II

5.1.8 Zusammenspiel der Schichten im ISO/ OSI- Modell

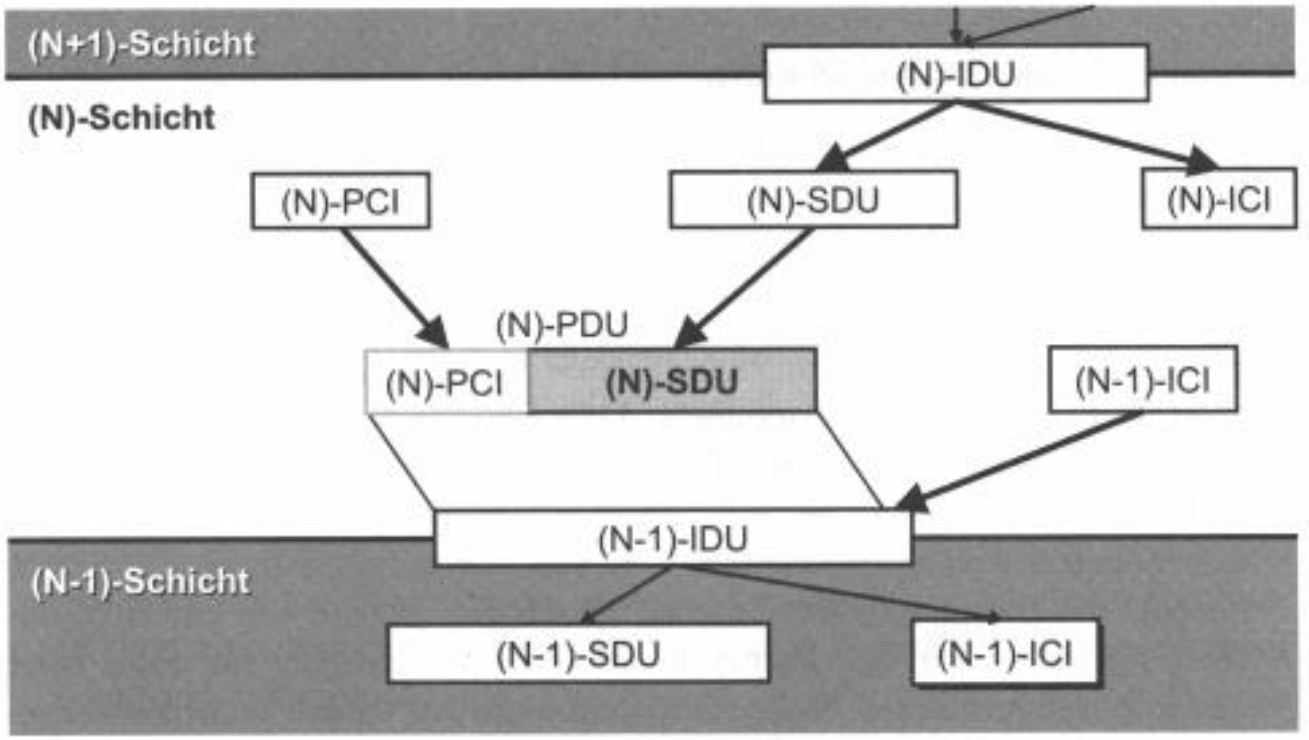


Abb. 4–10 *Zusammenspiel der Schichten*

IDU- Interface Data Unit; SDU- Service Data Unit, ICI- Interface Control Information, PCI- Protocol Control Unit, PDU- Protocol Data Unit

Rechnerkommunikation II

Datenkapselung

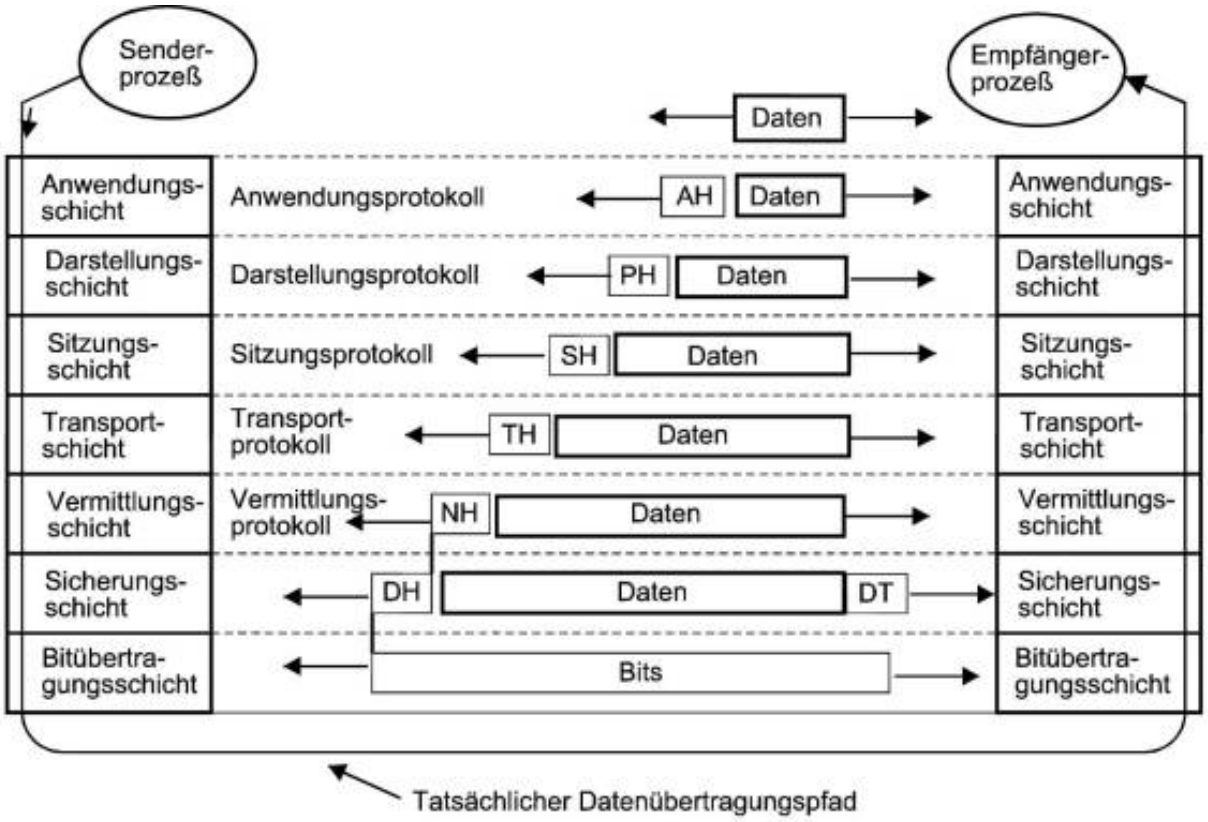


Abb. 1.17: Beispiel der Verwendung des OSI-Modells. Einige Header können Null sein (mit freundlicher Genehmigung von H.C. Folts)

Rechnerkommunikation II

Beispielablauf im OSI-Modell

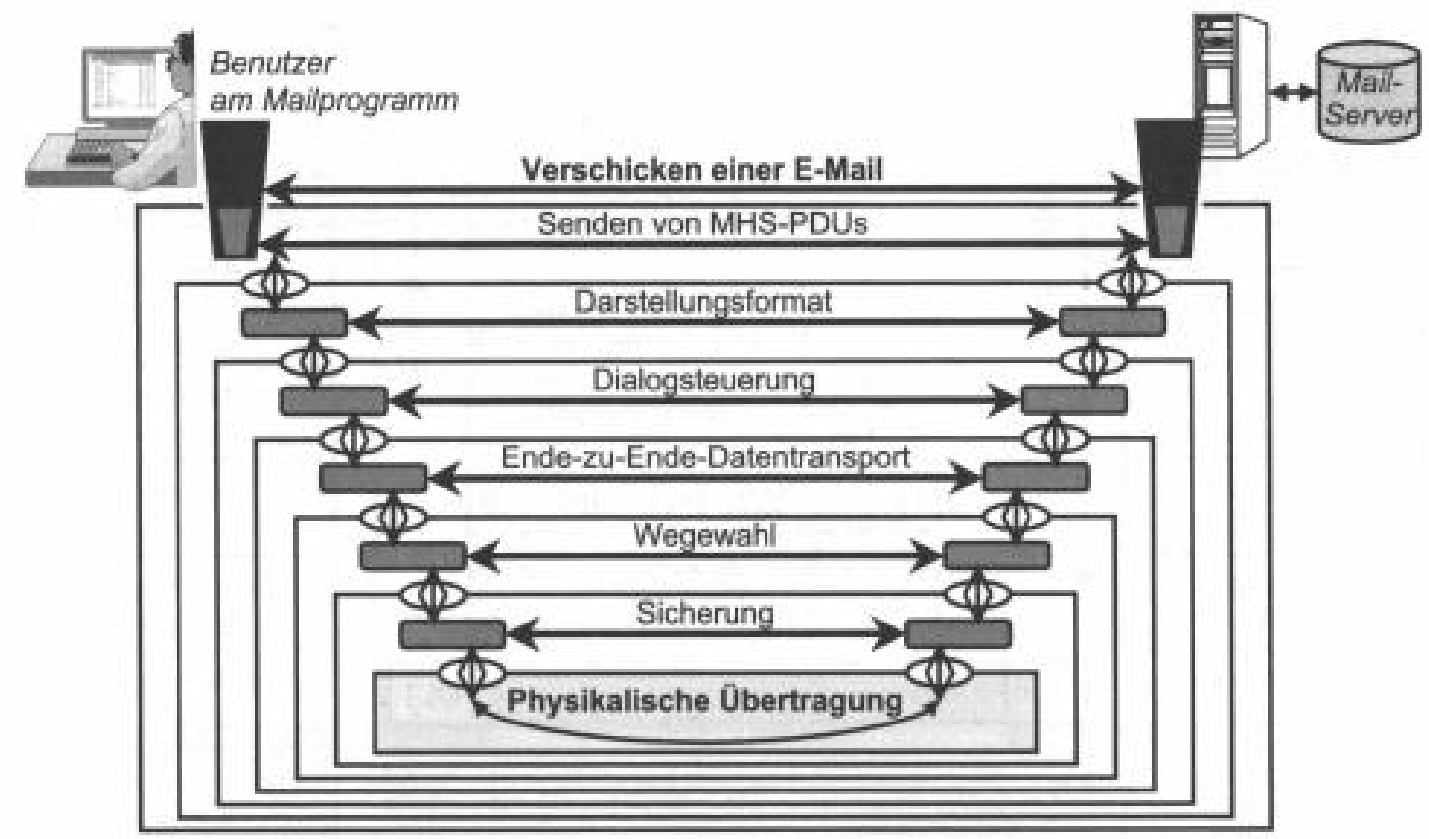


Abb. 4–12 *Beispielablauf – Übermittlung einer E-Mail*

(MSH- Message Handling System)

Rechnerkommunikation II

5.2 Internet-Kommunikationsarchitektur (TCP/ IP- Referenzmodell)

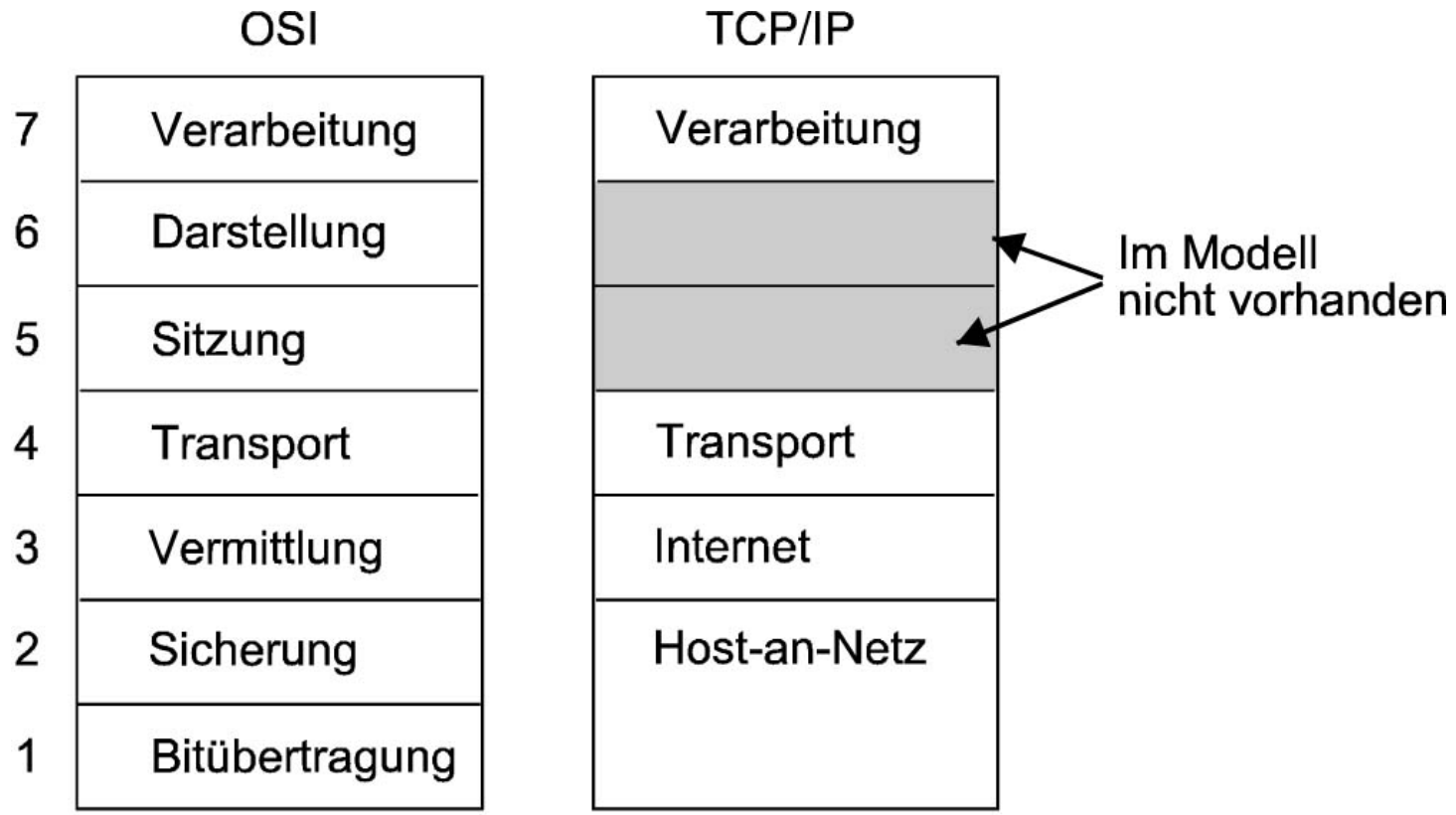


Abb. 1.18: Das TCP/IP-Referenzmodell

Rechnerkommunikation II

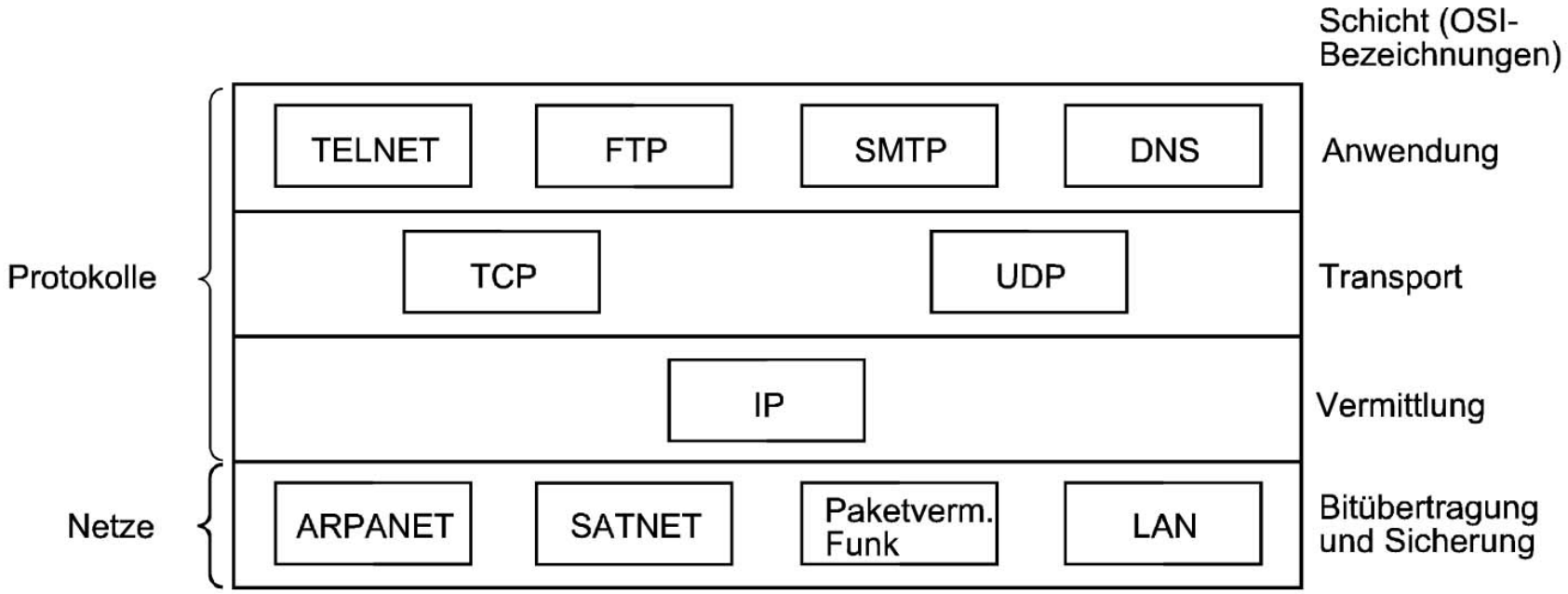


Abb. 1.19: Protokolle und Netze im ursprünglichen TCP/IP-Modell

Rechnerkommunikation II

4.2.1 Die Anwendungsschicht (Application Layer)

- Das TCP/ IP-Referenzmodell kennt im Gegensatz zum OSI-Modell keine Sitzungs- und Darstellungsschicht; sämtliche anwendungsbezogenen Vorgänge sind hier über die Protokolle und Dienste der Anwendungsschicht geregelt.
- z.B.: FTP, SMTP, DNS, HTTP

4.2.2 Die Transportschicht (Transport Layer)

- Aufgabe: Ermöglicht die Kommunikation zwischen gleichgestellten Einheiten auf Quell- und Zielhost (wie im OSI-Modell).
- Unterstützt im Gegensatz zum OSI-Modell auch den verbindungslosen Dienst
- Übertragungsprotokolle: TCP, UDP

4.2.3 Die Internetschicht (Internet Layer)

- Transport der Pakete über verschiedene Netze hinweg von Quell- zu Zielrechner; dabei spielt es für die übergeordnete Transportschicht keine Rolle ob die Pakete verschiedene Routen nehmen oder in falscher Reihenfolge ankommen → Routing
- unterstützt im Gegensatz zum OSI-Modell nur verbindungslosen Transport
- Überlastkontrolle
- Definition eines eigenen Paketformats und Protokolls: IP (Internet Protocol), IP-Paket
- entspricht weitgehend der Vermittlungsschicht im OSI-Modell

4.2.4 Host-zu-Netz-Schicht

Aufbau einer Verbindung über ein Protokoll;

→ Definitionslücke

Rechnerkommunikation II

4.3 Vergleich ISO/ OSI - TCP/ IP

- Das OSI-Modell unterscheidet genau zwischen den Konzepten Dienst (was macht die Schicht?), Protokoll (wie funktioniert die Schicht?) und Schnittstelle (wie kann auf die Dienste der Schicht unterhalb zugegriffen werden?).

Vergleich:

Konzepte OSI	OO- Konzepte
Dienst	Semantik von Methoden
Schnittstelle	Parameter und Ergebnisse von Methoden
Protokoll	Programmcode eines Objekts

Rechnerkommunikation II

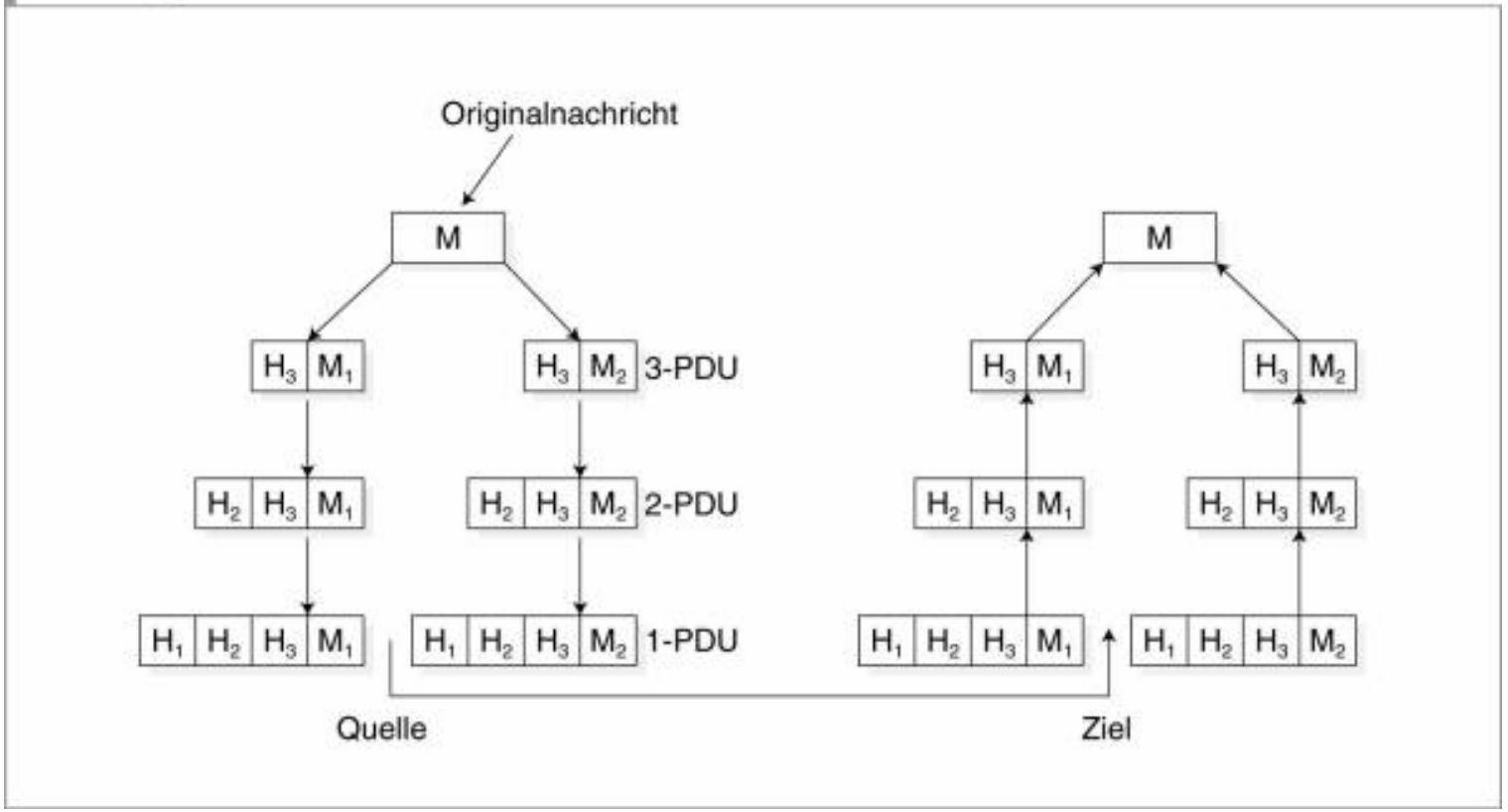
- Das TCP/ IP- Modell hingegen definiert die 4 vorhandenen Schichten nur über bestimmte Protokolle, im Falle der Host-zu-Netz-Schicht nur rudimentär. Die Bitübertragung und Sicherung der korrekten Übertragung stellen aber wichtige Aspekt dar.
- Das OSI- Modell ist ob seines 7-schichtigen Aufbaus relativ komplex → hohe Implementierungs- und Laufzeitkosten
- Das TCP/ IP- Modell wurde als Beschreibung eines vorhandenen Protokollstapels entwickelt. Daher eignet sich das Modell nur schlecht zur Beschreibung anderer Protokollstapel.

Rechnerkommunikation II

5. Das Internet - Protokolle und Schichten

5.1 PDUs und Schichten

Abbildung 1.23 Verschiedene PDUs auf unterschiedlichen Schichten der Protokollarchitektur



Rechnerkommunikation II

- Die ausgetauschten PDUs werden je nach Schicht besonders bezeichnet:

  Bitübertragungsschicht - Bit

  Sicherungsschicht - Frame

  Vermittlungsschicht/ Internetschicht - Datagram / Paket

  Transportschicht - Paket/ Segment

  Anwendungsschicht - Nachricht

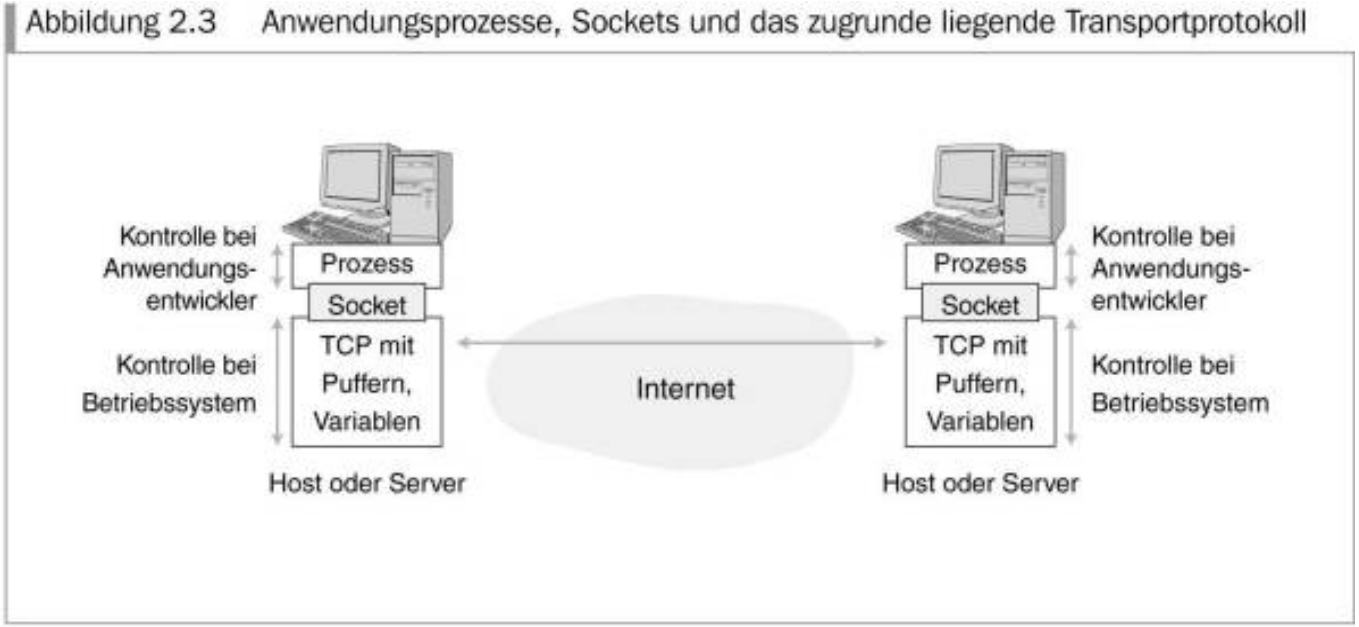
Rechnerkommunikation II

5.2 Protokolle der Anwendungsschicht

5.2.1 Kommunikation zwischen Anwendungsprozessen

- Ein Prozess kommuniziert mit einem anderen, indem er Nachrichten durch seinen *Socket* (allgem.: Kanal zum Zwecke des Informationsaustauschs) schickt.
- Der Socket leitet die Nachricht weiter an die Transportschicht.
- Der Prozess geht davon aus, dass auf die Empfängerseite über einen gleichen Socket verfügt, um die Nachricht zu empfangen.

Abbildung 2.3 Anwendungsprozesse, Sockets und das zugrunde liegende Transportprotokoll



Rechnerkommunikation II

5.2.2 Adressierung von Prozessen

- Um eine Nachricht von einem Prozess zu einem anderen zu senden, muss der empfangene Prozess identifiziert werden.
- Die Identifikation besteht aus zwei Teilen:
  - dem Namen oder der Adresse des Hosts,
  - einem Identifizierer, der den empfangenden Prozess auf dem Zielhost bezeichnet.
- Der Zielhost wird durch seine *IP-Adresse* identifiziert.
- Der Prozess wird durch eine *Portnummer* identifiziert.

Rechnerkommunikation II

5.2.3 Portnummern

- Beliebten Protokollen auf der Anwendungsschicht wurden spezifische Portnummern zugewiesen:
  - z.B.:

  Ein Web-Server-Prozess (der HTTP benutzt) wird durch Portnummer 80 identifiziert.

  Ein Mail-Server-Prozess (der SMTP benutzt) wird durch Portnummer 25 identifiziert.
- RFC 1700 enthält eine (veraltete) Liste der "wohlbekannten" Portnummern der Internet-Standardprotokolle. Die aktuelle Liste wird in einer Datenbank (www.iana.org) geführt.

Rechnerkommunikation II

5.2.4 Hypertext Transfer Protocol (HTTP)

5.2.4.1 Versionen

- HTTP 1.0 ist in RFC 1945 definiert.
- HTTP 1.1 ist in RFC 2616 definiert.
- HTTP/1.1 ist abwärtskompatibel zu HTTP/1.0.

5.2.4.2 Charakteristik von HTTP

- Protokoll für den Zugriff auf Webseiten
- HTTP unterstützt sowohl *persistente* Verbindungen als auch *nicht persistente* Verbindungen. HTTP/1.0 arbeitet mit nicht persistenten, HTTP/1.1 mit persistenten Verbindungen als Default-Modus.
- HTTP ist ein *zustandsloses* (kontextloses) Protokoll, d.h. der Server muss sich keine Kontextinfo hinsichtlich einer gestellten Anfrage merken (schließt die Verbindung mit der Antwort).
- HTTP ist ein pull-Protokoll, d.h. der Client zieht die Dateien von Server herunter.

5.2.4.3 Nicht persistente Verbindung

- Der HTTP-Client leitet eine TCP-Verbindung zum Server auf Port 80 als Default ein. Das Handshake findet statt.
- Der HTTP-Client schickt eine Anfragenachricht an den Server. Der Anfragename enthält bereit den Pfadnamen der gewünschten Datei.
- Der HTTP-Server empfängt die Anfrage, liest das angefragte Objekt ein, kapselt es in eine HTTP-Antwortnachricht und schickt diese an den Client.
- Der HTTP-Server weist TCP an, die Verbindung zu schließen, wenn der Client die Nachricht korrekt empfangen hat.
- Der HTTP-Client empfängt die Antwortnachricht, die TCP-Verbindung wird geschlossen. Der Client analysiert die Nachricht und sucht mögliche Referenzen in der HTML-Datei.
- Pro enthaltener Referenz werden die Schritte 1-4 wiederholt, bis alle Objekte vom Server zum Client übertragen wurden.

5.2.4.3 Persistente Verbindung

- Der Server läßt die TCP-Verbindung offen, nachdem er die Antwortnachricht verschickt hat.
- Anschließende Kommunikation zwischen Server und Client erfolgt auf der offenen TCP-Verbindung.
- Die TCP-Verbindung wird geschlossen, wenn sie für eine bestimmte (konfigurierbare) Zeitdauer nicht benutzt worden ist.
- Persistente Verbindungen gibt es mit und ohne *Pipelining*.
  - ohne Pipelining bedeutet, dass die TCP-Verbindung tatsächlich unbenutzt bleibt, solange der Client Daten empfängt. Der Client sendet nur dann eine neue Anfrage, wenn er die vorherige Antwort empfangen hat.
  - mit Pipelining bedeutet, dass der Client eine Anfrage nach einem neuen Objekt losschickt, sobald er auf eine Referenz stößt, ohne dass er die Antwort auf die vorherige Anfrage abwartet.
  - HTTP/1.1 nutzt im Default-Modus persistente Verbindungen mit Pipelining.

Rechnerkommunikation II

5.2.4.4 Methoden

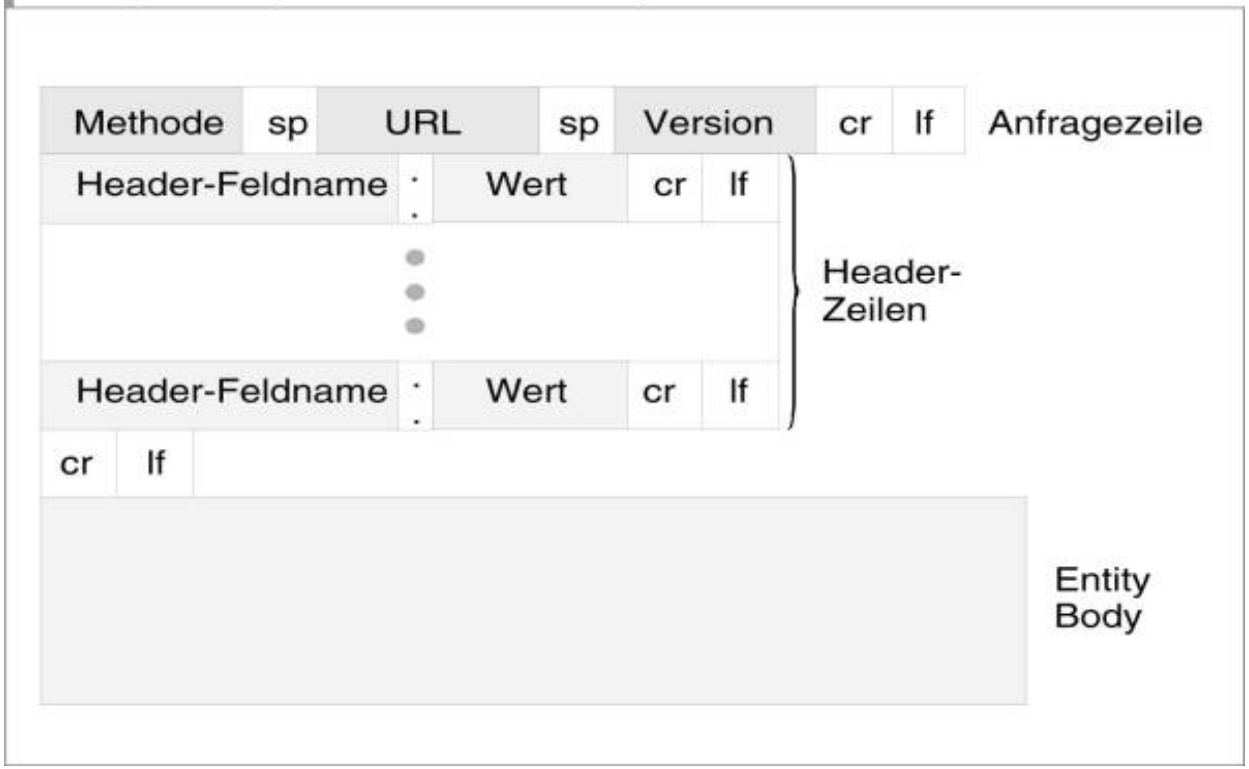
- Methoden spezifizieren eine Anfrage. Die am häufigsten verwendete Anfrage ist die Anforderung einer Webseite: GET

Methode	Beschreibung
GET	Anforderung zum Lesen einer Webseite
HEAD	Anforderung zum Lesen des Headers einer Webseite
PUT	Anforderung zum Speichern einer Webseite
POST	Anhängen an eine benannte Ressource (z. B. eine Webseite)
DELETE	Entfernen einer Webseite
TRACE	Echo für die eingehende Anforderung
CONNECT	Für zukünftige Verwendung reserviert
OPTIONS	Abfrage bestimmter Optionen

5.2.4.5 Formatstruktur einer Anfrage

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language:fr
(data data …)
```

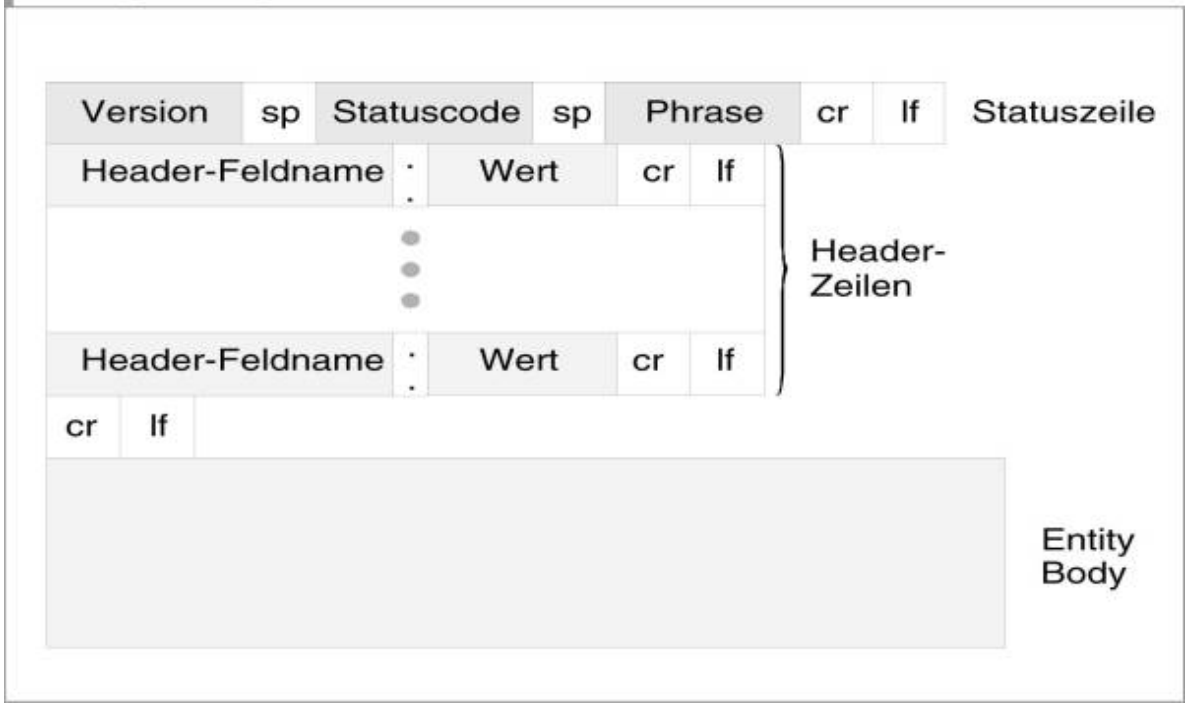
Abbildung 2.7 Allgemeines Format einer Anfragenachricht



5.2.4.6 Formatstruktur einer Antwort

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 09:23:24GMT
Content-Length: 6821
Content-Type: text/html
(data data data data data . . .)
```

Abbildung 2.8 Allgemeines Format einer Antwortnachricht



Rechnerkommunikation II

5.2.4.7 Statuscodes

Der Statuscode der Antwortnachricht spezifiziert die Antwort:

Code	Bedeutung	Beispiel
1xx	Information	100 = Server stimmt zu, die Anforderung des Clients zu bearbeiten
2xx	Erfolg	Anforderung erfolgreich; 204 = kein Inhalt vorhanden
3xx	Umleitung	301 = Seite verzogen; 304 = Seite im Cache noch gültig
4xx	Client-Fehler	403 = verbotene Seite, 404 = Seite nicht gefunden
5xx	Server-Fehler	500 = interner Server-Fehler; 503 = versuche es später noch einmal

5.2.4.8 Nachrichten-Header

Sowohl Anforderung als auch Antwort können einen Header haben, der zusätzliche Informationen aufnimmt:

Header	Typ	Inhalte
User-Agent	Anforderung	Information über den Browser und dessen Plattform
Accept	Anforderung	Seitentypen, die der Client verarbeiten kann
Accept-Charset	Anforderung	Zeichensätze, die der Client unterstützt
Accept-Encoding	Anforderung	Seitenkodierungen, die der Client verarbeiten kann
Accept-Language	Anforderung	Natürliche Sprachen, die der Client verarbeiten kann
Host	Anforderung	DNS-Name des Servers
Authorization	Anforderung	Eine Liste der Anmeldeinformationen des Clients
Cookie	Anforderung	Sendet ein zuvor gesendetes Cookie an den Server zurück
Date	Beides	Datum und Uhrzeit, zu der die Nachricht gesendet wurde
Upgrade	Beides	Das Protokoll, auf das der Sender wechseln möchte
Server	Antwort	Informationen über den Server
Content-Encoding	Antwort	Wie der Inhalt kodiert ist (z. B. gzip)
Content-Language	Antwort	Natürliche Sprache der Seite
Content-Length	Antwort	Seitenlänge in Byte
Content-Type	Antwort	MIME-Typ der Seite
Last-Modified	Antwort	Zeit und Datum, an dem die Seite zuletzt geändert wurde
Location	Antwort	Ein Befehl an den Client, seine Anforderung an einen anderen Ort zu senden
Accept-Ranges	Antwort	Der Server akzeptiert Byte-Bereichsanfragen
Set-Cookie	Antwort	Der Server möchte, dass der Client ein Cookie speichert

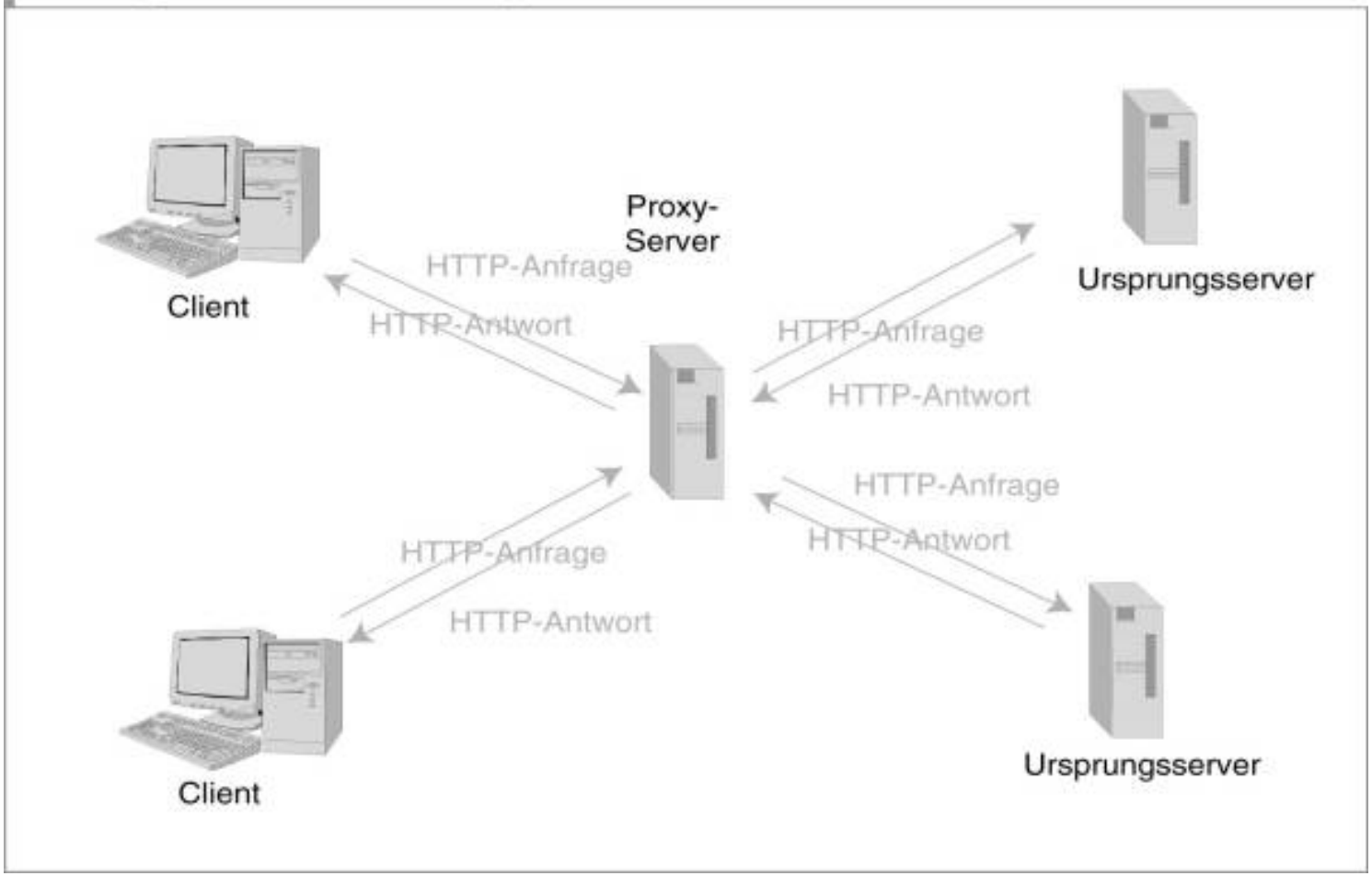
Rechnerkommunikation II

SW: Web-Cache

- Ein Web-Cache, auch *Proxy-Server* genannt, ist eine Netzwerkeinheit, die HTTP-Anfragen im Auftrage eines Client erfüllt.
- Der Client schickt eine Anfragenachricht zuerst an den Web-Cache.
- Der Web-Cache prüft, ob das geforderte Objekt in seinem Speicher vorrätig und nicht veraltet ist.
- Wenn ja, schickt er dem Client das Objekt, ohne den Originalserver zu kontaktieren. Wenn nein, wird der Web-Cache zum Client und fragt beim Originalserver das Objekt an.

Rechnerkommunikation II

Abbildung 2.9 Clients fordern Objekte über einen Web-Cache an.



Rechnerkommunikation II

Aufgaben

1) **Wiederholen** Sie den Stoff dieser Sitzung **bis zur nächsten Sitzung** (siehe dazu den Link zur Sitzung auf der HKI-Homepage).
   Informieren Sie sich zusätzlich durch eigene Literaturrecherche!

2) Beantworten Sie die Fragen aus der Sammlung **„beispielhafte Klausurfragen“** zur Rechnerkommunikation (soweit in dieser Sitzung behandelt).