Grundlagen Rechnernetze und Verteilte Systeme

SoSe 2014

Kapitel 5: Sitzungs-, Darstellungs- und Anwendungsschicht

Prof. Dr.-Ing. Georg Carle

Nadine Herold, M. Sc.
Dipl.-Inf. Stephan Posselt
Johannes Naab, M. Sc.
Marcel von Maltitz, M. Sc.
Stephan Günther, M. Sc.

Fakultät für Informatik
Lehrstuhl für Netzarchitekturen und Netzdienste
Technische Universität München

Worum geht es in diesem Kapitel?

1 Motivation

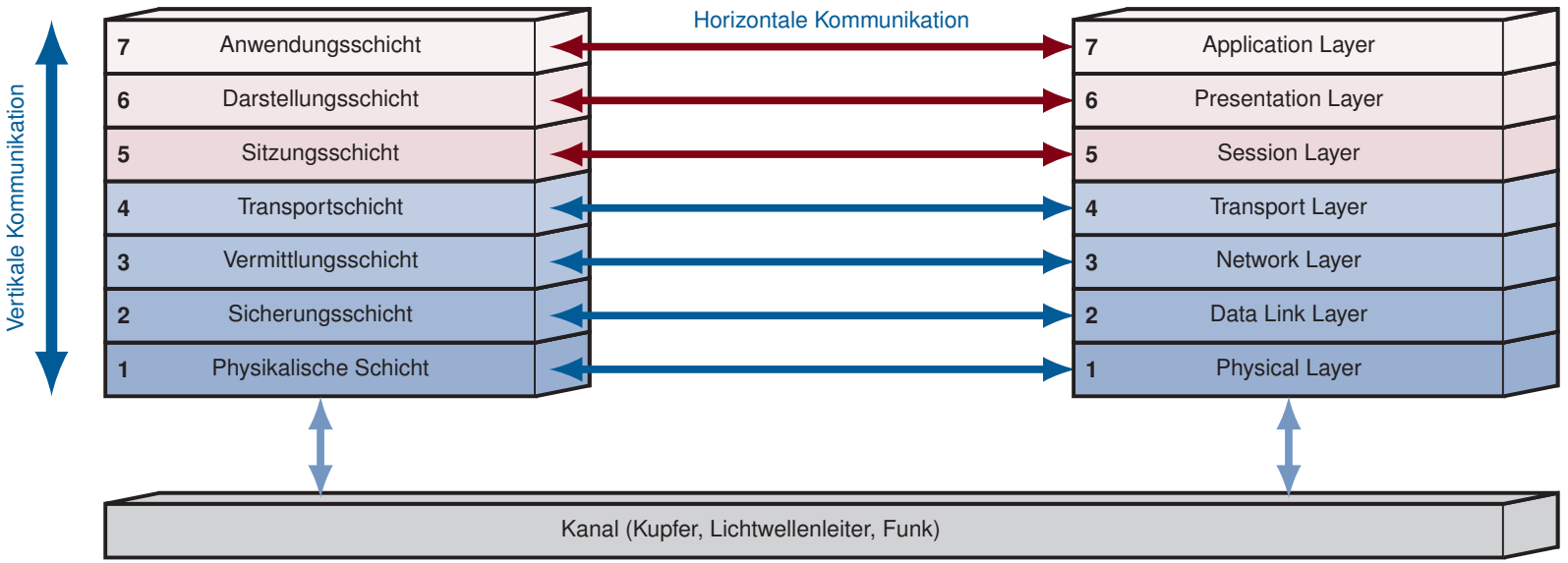
2 Sitzungsschicht

3 Darstellungsschicht

4 Anwendungsschicht

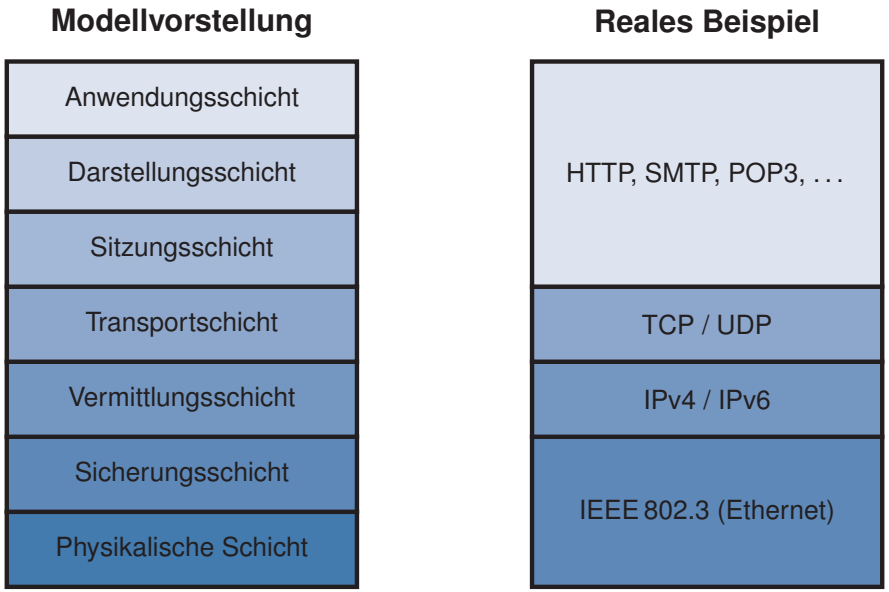
5 Zusammenfassung

Einordnung im ISO/OSI-Modell



Modell und Realität

- Dienste der Sitzungs- und der Darstellungsschicht sind in einzelnen Fällen in Form standardisierter Protokolle implementiert.
- In anderen Fällen sind Funktionen, die der Sitzungs- bzw. der Darstellungsschicht zuzuordnen sind, in die Anwendung integriert.



Modell und Realität

- Die Standards der ITU-Serie X.200 beschreiben Dienste der sieben OSI-Schichten, sowie Protokolle zur Erbringung dieser Dienste.
- Die in diesen Standards vorgenommene Strukturierung ist nützlich.
- Etliche OSI-Protokolle haben in der Praxis kaum Bedeutung.

Im Folgenden werden wir

- die Aufgaben der Sitzungs- und Darstellungsschicht erläutern,
- beispielhaft einige Protokolle kennenlernen, welche den Schichten 5 und 6 zugeordnet werden können,
- sowie wichtige Protokolle der Anwendungsschicht erläutern.

Übersicht

Sitzungsschicht

Die Sitzungsschicht kann nach X.200 in zwei verschiedenen **Modi** betrieben werden:

- **Connection-Oriented Mode:**
  Die Sitzungsschicht baut eine Verbindung zwischen den Kommunikationspartnern auf, die über die Dauer einzelner Datentransfers hinweg erhalten bleibt.
  Es werden die Phasen Verbindungsaufbau, Datentransfer und Verbindungsabbau unterschieden.
  Die Sitzungsschicht kann in diesem Fall vielfältige Aufgaben erfüllen.
  Die unterstützte Funktionalität beinhaltet die Koordination mehrerer beteiligter Parteien bezüglich Datenfluss, verwendeter Dienste, Aushandlung diverser Parameter für den Kommunikationsablauf und Identifikation der Verbindungen.

- **Connectionless Mode:**
  In diesem Fall wird durch die Sitzungsschicht nur ein sehr einfacher Dienst erbracht.
  Von der dienstnehmenden Schicht werden die zu übertragenden Daten entgegengenommen, zusammen mit Informationen zur Adressierung und zu den Dienstgüteanforderungen, und an die Transportschicht weitergereicht.

Eine Verbindung der Sitzungsschicht ist nicht gleichbedeutend mit einer Verbindung der Transportschicht! Eine Session kann beispielsweise nacheinander mehrere TCP-Verbindungen beinhalten.

Dienste der Sitzungsschicht

Definition (Session)

Eine Session beschreibt die Kommunikation zwischen zwei oder mehreren Teilnehmern, mit definiertem Anfang und Ende und sich daraus ergebender Dauer.

Um für die dienstnehmende Schicht (Darstellungsschicht) eine Dialogführung zu ermöglichen, müssen gegebenenfalls mehrere Transportschicht-Verbindungen verwendet und kontrolliert werden. Dies kann auch die Behandlung abgebrochener und wiederaufgenommer TCP-Verbindungen beinhalten.
Im Connection-Oriented Mode werden verschiedene Dienste angeboten:

- **Aufbau** und **Abbau** von Sessions,
- normaler und beschleunigter **Datentransfer** [1],
- Token-Management zur **Koordination** der Teilnehmer,
- **Synchronisation** und Resynchronisation,
- **Fehlermeldungen** und Aktivitätsmanagement, sowie
- **Erhaltung** und **Wiederaufnahme** von Sessions nach Verbindungsabbrüchen.

Die Sitzungsschicht stellt im Connectionless Mode folgenden einfachen Dienst bereit:

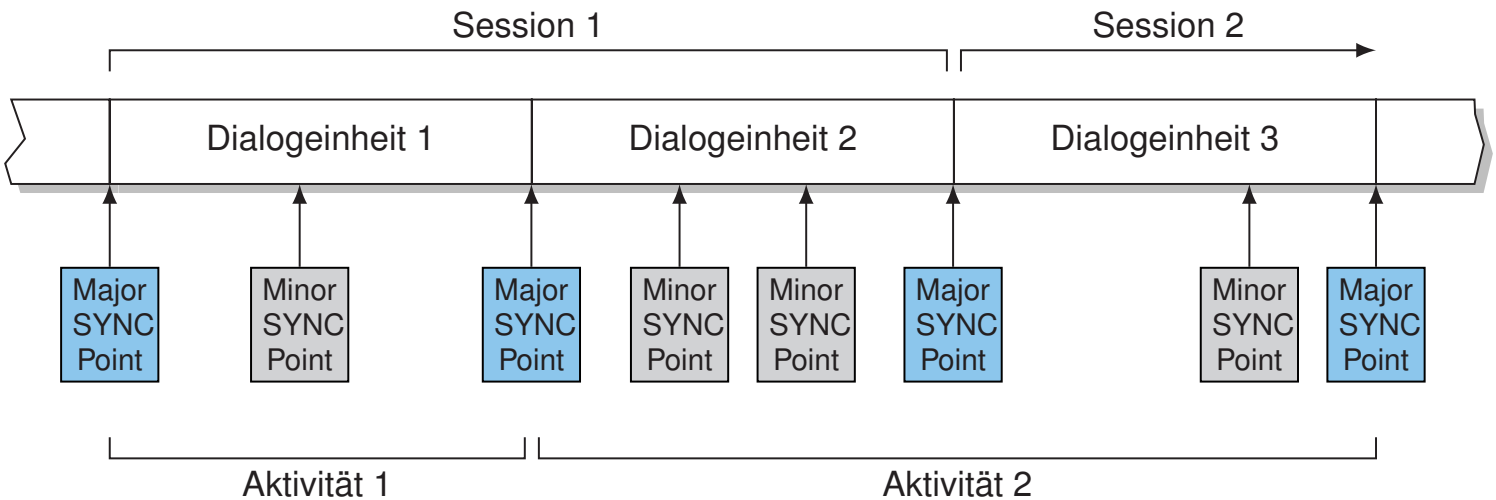
- **Datentransfer**

[1] Expedited Data Transfer: dringliche Daten, wie Alarme oder Interrupts

Funktionseinheiten der Sitzungsschicht

Die Dienste der Sitzungsschicht werden gemäß dem Standard X.215 in Funktionseinheiten gruppiert:

Bezeichnung	Beschreibung
Kernel	Bereitstellung von Basisfunktionen
Halb-duplex	Abwechselndes Senden und Empfangen
Duplex	Gleichzeitiges Senden und Empfangen
Negotiated Release	Beenden der Session mit gegenseitigem Bestätigen
Expedit Data	Beschleunigter Datentransfer
Activity Management	Logische Strukturierung der Session
Major Synchronization	Interne Strukturierung der Sessions in Dialogeinheiten
Minor Synchronization	Interne Strukturierung der Dialogeinheiten



Kombination von Funktionseinheiten

Unterschied zwischen Aktivitätsmanagement und Synchronisation

Aktivitäten bestehen aus mehreren Dialogeinheiten und können jederzeit unterbrochen und später (in der gleichen oder auch einer anderen Sitzung) wieder aufgenommen werden; d.h. eine Aktivität kann auch über mehrere Sessions hinweg existieren.
Synchronisation erfolgt durch sog. Synchronisationspunkte, an welchen Sessions oder Aktivitäten wieder aufgesetzt werden können, oder an welchen eine Resynchronisation (Zurücksetzen) möglich ist. Die Kommunikationspartner können hier prüfen, wie weit die Kommunikation, z. B. eine Datenübertragung, vorangeschritten ist.

Folgende Kombinationen von Funktionseinheiten sind vorgesehen:

- BCS Basic Combined Subset
  Kernel, Halb-Duplex/Duplex
- BSS Basic Synchronized Subset
  Kernel, Halb-Duplex, Negotiated Release, Minor/Major synchronize, Resynchronize
- BAS Basic Activity Subset
  Kernel, Halb-Duplex, Minor sychronize, Execptions, Activity Management

Eine gültige Kombination wird vor Beginn der Session zwischen den Kommunikationspartnern ausgehandelt.

Synchronisation

Es werden folgende Arten von Synchronisationspunkten unterschieden:

- Major Synchronisationspunkte
- Minor Synchronisationspunkte

Definition (Major und Minor Synchronisationspunkte)

Major Synchronisationspunkte werden verwendet, um die Struktur der ausgetauschten Daten in eine Serie von Dialogeinheiten zu zerlegen. Sie müssen explizit bestätigt werden.

Minor Synchronisationspunkte werden für die Strukturierung innerhalb dieser Dialogeinheiten verwendet. Sie können bestätigt werden.
Bis zu einem solchen Punkt versandte Daten werden nicht von einem Resynchronisationsprozess verworfen.

Um die Kommunikation zu steuern, werden Marken (Token) verwendet:

- Datenmarken (“data token” - geben an, wer bei der Verwendung des Halb-duplex-Betriebs senden darf)
- Beendigungsmarken (“release token” - beenden eine Sitzung)
- Synchronisationsmarken (“synchronize-minor token”)
- Aktivitätsmarken (“activity-major token”)

Quality of Service im Session Layer

Auf der Sitzungsschicht kann zwischen verschiedenen QoS-Parametern unterschieden werden:

- Service Parameter
- Performance Parameter

Es werden zunächst die verschiedenen Service Parameter erläutert.

Definition (Service Parameter)

Protection: Beschreibt den Schutz gegen unautorisiertes Lesen oder Manipulation von Nutzerinformationen

Priorität: Beschreibt die Relation zwischen den einzelnen Sessions (Aufteilung von Ressourcen)

Resilience: Beschreibt die Wahrscheinlichkeit, dass eine Session ordnungsgemäß durchgeführt werden kann bzw. wie viele Fehler tolieriert werden können

Performance Parameter des Session Layers

- Aufbau der Session: Establishment Delay und Establishment Failure Probability
- Datentransfer
  - Durchsatz und Transit Delay
  - Residual Error Rate und Transfer Failure Probability
- Abbau der Session: Release Delay und Release Failure Probability

Definition (Residual Error Rate)

Die RER bezeichnet das Verhältnis der Menge fehlerhaft übertragener, verlorener oder dupliziert empfangener Daten zur Gesamtmenge der gesendeten Daten:

$$\text{RER} = \frac{S_e + S_l + S_x}{S}.$$

- S_e beschreibt fehlerhaft übertragene Daten, S_l verloren gegangene Daten und S_x dupliziert empfangenen Daten.
- S beschreibt die Gesamtmenge gesendeter Daten.
- Die Gesamtmenge S gesendeter Daten setzt sich zusammen aus $S_s + S_e + S_l + S_x$, mit der Menge korrekt übertragener Daten S_s.
- Die Menge empfangener Daten setzt sich zusammen aus $S_s + S_e + S_x$.

Übersicht

Darstellungsschicht

Die Aufgabe der Darstellungsschicht (engl. Presentation Layer) ist es, den Kommunikationspartnern eine einheitliche Interpretation der Daten zu ermöglichen, d. h. Daten in einem einheitlichen Format zu übertragen.

Der Darstellungsschicht sind grundsätzlich folgende Aufgaben zugeordnet:

- die Darstellung der Daten (Syntax),
- die Datenstrukturen zur Übertragung der Daten
- die Darstellung der Aktionen an diesen Datenstrukturen, sowie
- Datentransformationen.

Die Darstellung auf Schicht 6 muss nicht der Darstellung auf Schicht 7 (Anwendungsschicht) entsprechen. Die Darstellungsschicht ist für die Syntax der Nutzdaten verantwortlich, die Semantik verbleibt bei den Anwendungen.

Unter Syntax versteht man die Darstellung von Daten nach bestimmten Regeln (Grammatik). Werden Daten durch Bedeutung ergänzt, spricht man von Informationen, dies ist die Aufgabe der Semantik.

- Anwendungen sollen syntaxunabhängig miteinander kommunizieren können.
- Anwendungsspezifische Syntax kann von der Darstellungsschicht in eine einheitliche Form umgewandelt und dann übertragen werden.

Aufgaben der Darstellungsschicht

Den grundsätzlichen Aufgaben der Darstellungsschicht lassen sich konkrete Funktionen zuordnen:

- Quellenkodierung und Datenkompression
- Umkodierungen und Übersetzung zwischen Datenformaten
- Strukturierte Darstellung von Informationen
- Ver- und Entschlüsselung

Existierende Protokolle lassen sich nicht immer eindeutig einer Schicht zuordnen. Relevante Protokolle (wie beispielsweise TLS) beinhalten sowohl Funktionen, die üblicherweise der Schicht 5 zugeordnet werden, als auch Funktionen, die üblicherweise der Schicht 6 zugeordnet werden.

Frage: Warum lässt sich das TLS-Protokoll sowohl der Schicht 5 als auch der Schicht 6 zuordnen?

Aufgaben der Darstellungsschicht

Den grundsätzlichen Aufgaben der Darstellungsschicht lassen sich konkrete Funktionen zuordnen:

- Quellenkodierung und Datenkompression
- Umkodierungen und Übersetzung zwischen Datenformaten
- Strukturierte Darstellung von Informationen
- Ver- und Entschlüsselung

Existierende Protokolle lassen sich nicht immer eindeutig einer Schicht zuordnen. Relevante Protokolle (wie beispielsweise TLS) beinhalten sowohl Funktionen, die üblicherweise der Schicht 5 zugeordnet werden, als auch Funktionen, die üblicherweise der Schicht 6 zugeordnet werden.

Frage: Warum lässt sich das TLS-Protokoll sowohl der Schicht 5 als auch der Schicht 6 zuordnen?

- Der TLS Handshake dient dem Aushandeln von Sitzungsschlüssel (engl. Session Keys).
- Dies entspricht dem Aushandeln der Kommunikationsparameter einer Sitzung und kann somit der Sitzungsschicht zugeordnet werden.
- Bei den Funktionen Kompression sowie Ver- und Entschlüsseln von Nutzdaten handelt es sich um Funktionen, die der Darstellungsschicht zugeordnet werden.
- TLS lässt sich somit sowohl der Schicht 5 als auch der Schicht 6 zuordnen.

Datenkompression und Umkodierung

Datenkompression und Umkodierung gehört zu den wichtigsten Funktionen der Darstellungsschicht. Es ist wichtig, die zugrundeliegenden Prozessen zu unterscheiden.

Definition (Umkodierung)

Umkodierung oder Transkodierung von Daten beschreibt den Prozess der Überführung von einer Darstellung in eine andere. Hierbei sind keine Einschränkungen wie bezüglich der Anzahl verwendeter Zeichen o. ä. nötig. Eine gültige Umkodierung kann gegebenen Nutzdaten in ihrem Umfang auch erweitern.

Definition (Kompression)

Unter Datenkompression versteht man die Entfernung von Redundanz. Die komprimierte Nachricht ist i. A. kürzer als das Original.

- Wir wollen im Folgenden näher auf Kompression eingehen.
- Als Beispiel für ein Kompressionsverfahren werden wir den Huffman-Algorithmus betrachten.
- Zudem wollen wir wichtige Eigenschaften des Huffman-Algorithmus näher analysieren.

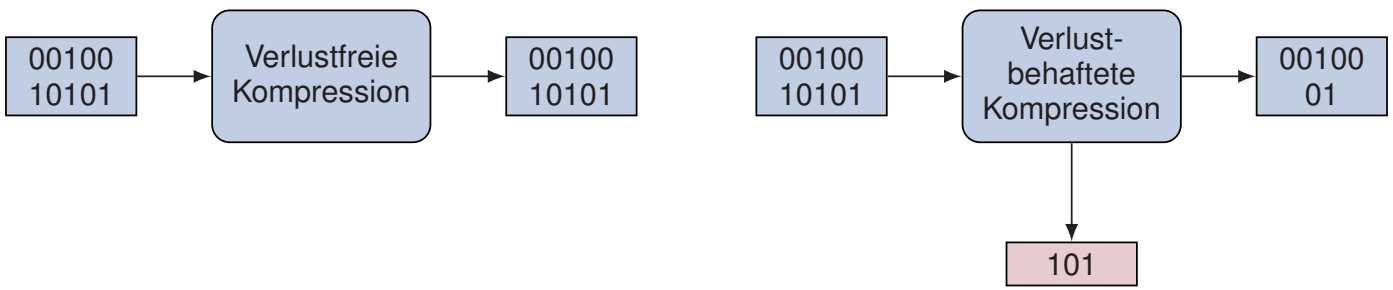
Einteilung von Datenkompression und Umkodierung

Kodierungsverfahren können unterschieden werden nach

- Fixed-Length Code bei welchen alle Zeichen mit der gleichen Anzahl Bits kodiert werden, z.B. ASCII (8 Bits) oder UNICODE (16 Bits).
- Variable-Length Code bei welchen alle Zeichen abhängig von ihrer Auftrittswahrscheinlichkeit mit einer Kodierungslänge korreliert werden.

Es lassen sich zwei grundlegende Arten der Kompression unterscheiden:

- Verlustfreie Komprimierung (engl. Lossless Data Compression): Die dekodierten Daten können identisch zum Original wieder hergestellt werden, ohne dass Informationen verloren gehen oder verändert werden (wie z.B. beim ZIP-Dateiformat, das den Einsatz unterschiedlicher verlustfreier Kompressionsalgorithmen ermöglicht).
- Verlustbehaftete Komprimierung (engl. Lossy Data Compression): Bei der Dekodierung gehen Informationen der ursprünglichen Daten verloren und können nicht mehr vollständig richtig dekodiert werden (siehe z.B. das verlustbehaftete Kompressionsverfahren JPEG).



Beispiel: Kompression mittels Huffman-Codes

- Viele Protokolle komprimieren Daten vor dem Senden (Quellenkodierung).
- TLS beispielsweise bietet optional Kompressionsmethoden. Diese werden vor der Verschlüsselung angewandt. (Warum davor?)
- Ein häufig (u. a. von TLS) verwendetes Kompressionsverfahren für Texte ist der Huffman-Code.

Grundlegende Idee der Huffman-Kodierung:

- Nicht alle Textzeichen treten mit derselben Häufigkeit auf, z. B. tritt der Buchstabe „E“ in der deutschen Sprache mit einer Häufigkeit von 17.4 % gefolgt von „N“ mit 9.78 % auf.
- Anstelle Zeichen mit uniformer Codewortlänge zu kodieren, (z. B. ASCII-Code), weise häufigen Zeichen kürzere Codewörter zu.

Bei der Huffman-Kodierung handelt es sich also um ein

- verlustfreies Kompressionsverfahren.

Konstruktion eines Huffman-Codes:

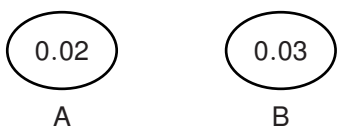
Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$. Es wird außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30

Konstruktion eines Huffman-Codes:

Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$. Es wird außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

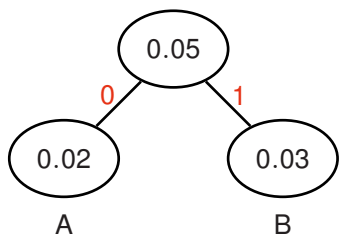
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Konstruktion eines Huffman-Codes:

Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$. Es wird außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

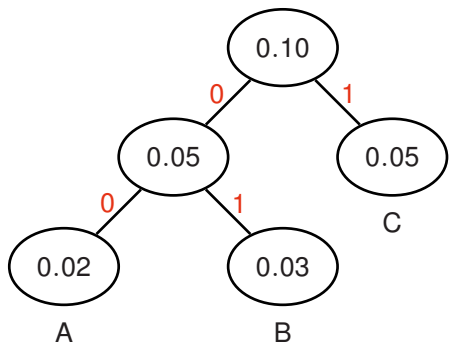
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Konstruktion eines Huffman-Codes:

Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$. Es wird außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

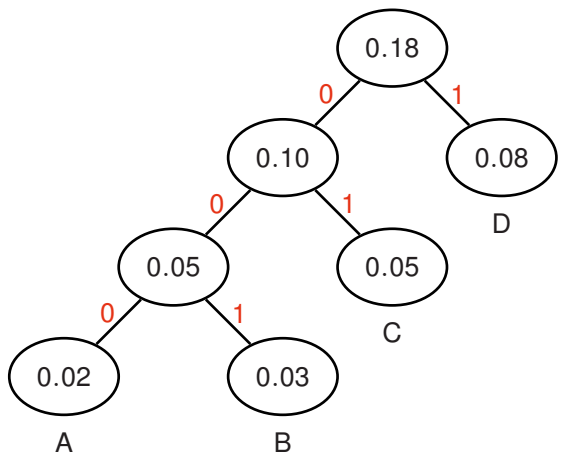
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Konstruktion eines Huffman-Codes:

Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$. Es wird außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

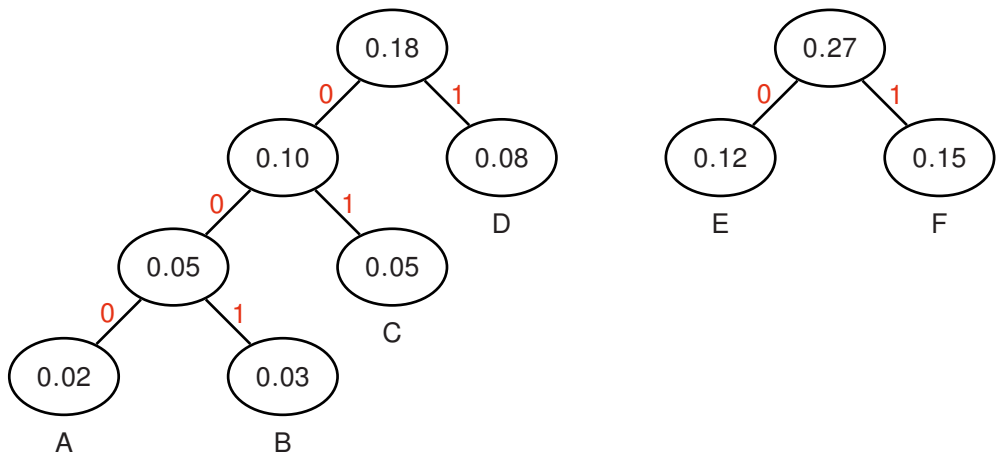
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Konstruktion eines Huffman-Codes:

Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$. Es wird außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

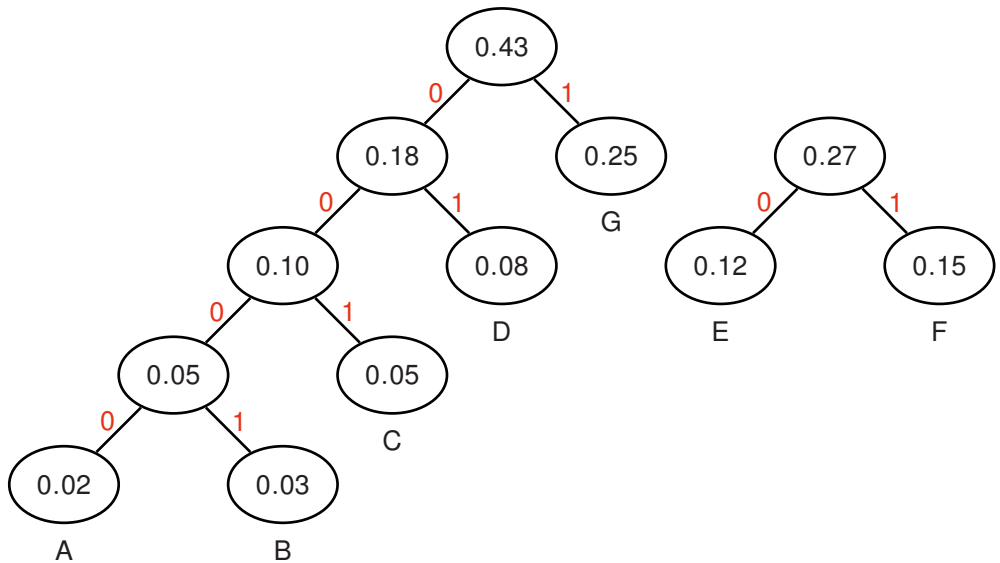
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Konstruktion eines Huffman-Codes:

Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$. Es wird außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

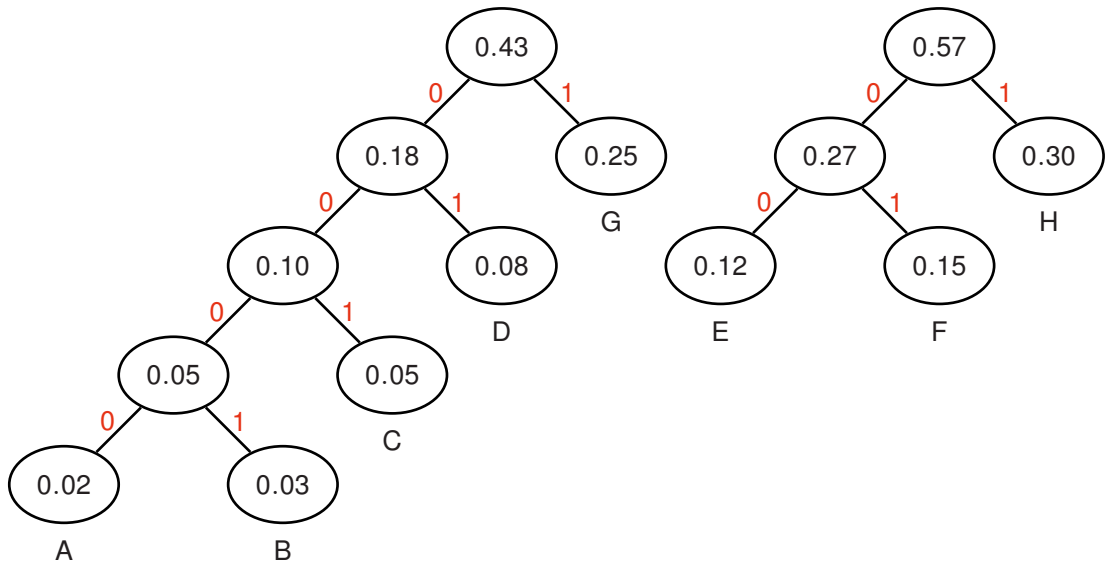
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Konstruktion eines Huffman-Codes:

Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$. Es wird außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

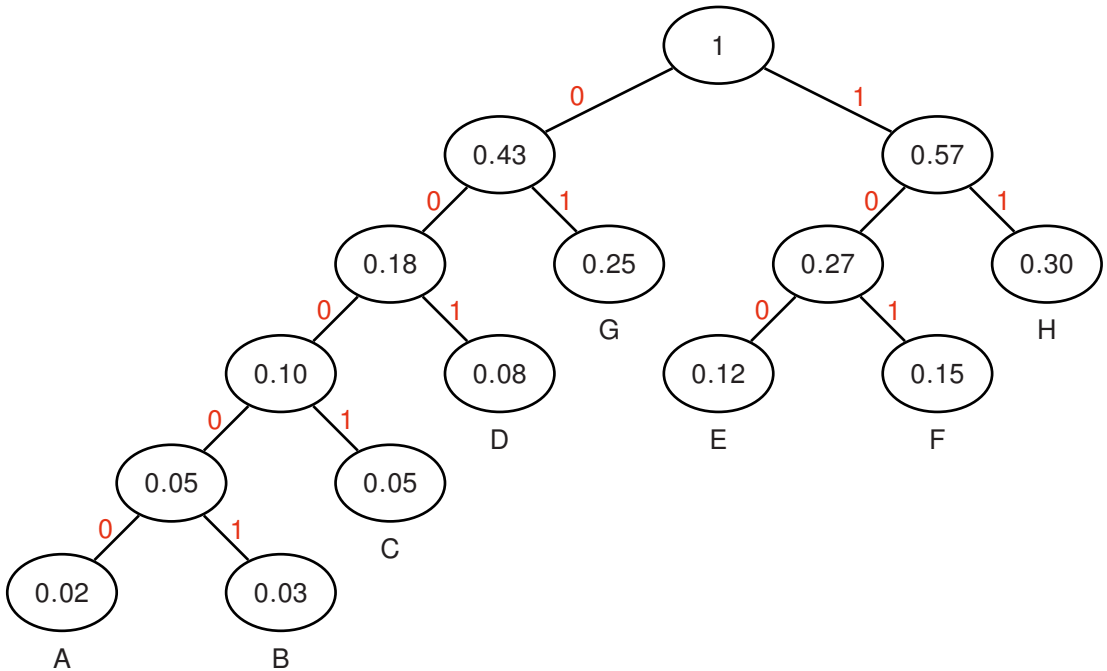
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Konstruktion eines Huffman-Codes:

Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$. Es wird außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

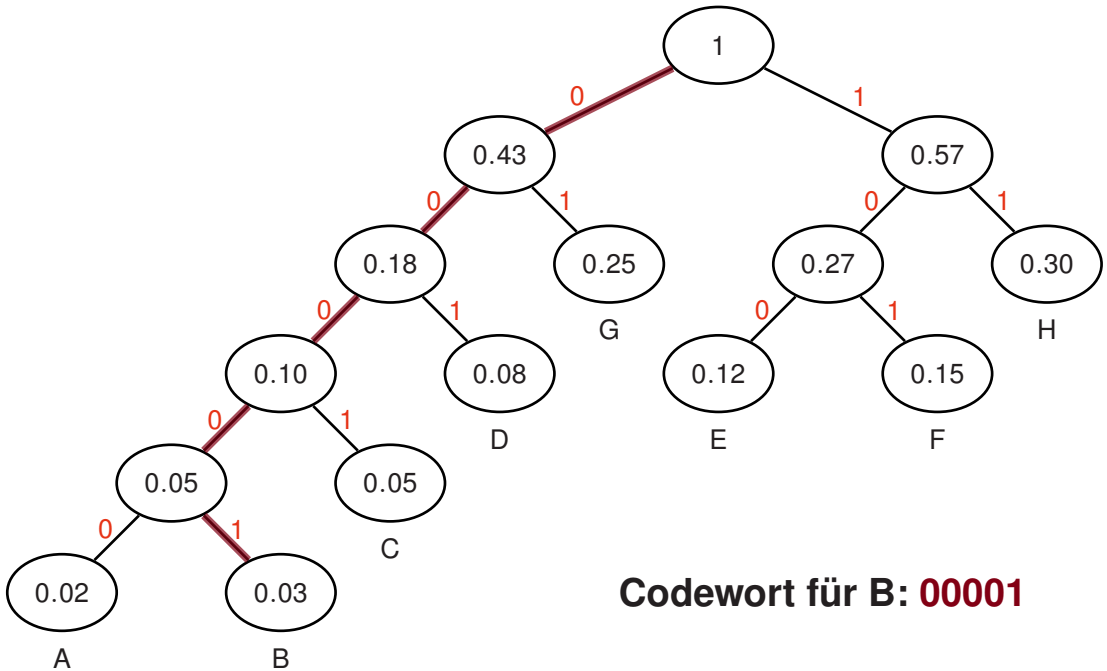
z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Konstruktion eines Huffman-Codes:

Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$. Es wird außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

z	$\Pr[X = z]$
A	0.02
B	0.03
C	0.05
D	0.08
E	0.12
F	0.15
G	0.25
H	0.30



Codewort für B: 00001

Durchschnittliche Codewortlänge

z	$\Pr[X = z]$	Huffman-Code	Länge $l_H(z)$	„Einfacher“ Code
A	0.02	00000	5	000
B	0.03	00001	5	001
C	0.05	0001	4	010
D	0.08	001	3	011
E	0.12	010	3	100
F	0.15	011	3	101
G	0.25	10	2	110
H	0.30	11	2	111

- Uniformer Code:

$$\mathrm{E}[l(z)] = 3.0, \quad \text{da alle Codewörter gleich lang sind}$$

- Huffman-Code:

$$\mathrm{E}[l_H(z)] = \sum_{z \in \mathcal{A}} \Pr[X = z] \cdot l_H(z) = 2.6$$

$$\Rightarrow \text{Die Einsparung beträgt } 1 - \mathrm{E}[l_H(z)]/\mathrm{E}[l(z)] \approx 13\,\%$$

Anmerkungen zum Huffman-Code:

- Statische Huffman-Codes sind darauf angewiesen, dass die Auftrittswahrscheinlichkeit der Zeichen den Erwartungen entspricht.
- Zeichenhäufigkeiten können dynamisch bestimmt werden, allerdings muss dem Empfänger dann das verwendete Codebuch mitgeteilt werden.
- Längere Codewörter (z. B. ganze Wörter statt einzelner Zeichen) werden infolge der Komplexität zum Bestimmen des Codebuchs ein Problem.
- Der Huffman-Code ist ein optimaler und präfixfreier Code.

Definition (Optimaler Präfixcode)

Bei einem Präfixfreien Code sind gültige Codewörter niemals Präfix eines anderen Codeworts desselben Codes. Ein optimaler Präfixfreier Code minimiert darüber hinaus die mittlere Codewortlänge

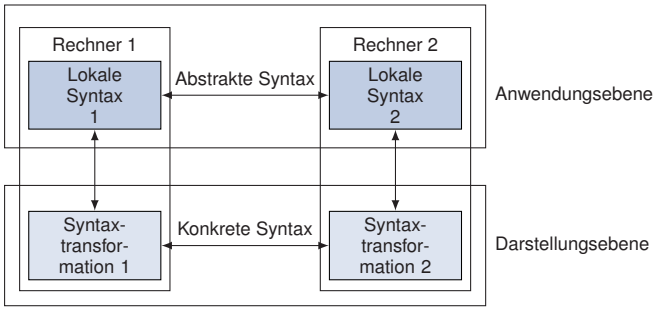
$$\sum_{i \in \mathcal{A}} p(i) \cdot |c(i)|,$$

wobei $p(i)$ die Auftrittswahrscheinlichkeit von $i \in \mathcal{A}$ und $c(i)$ die Abbildung auf ein entsprechendes Codewort bezeichnen.

- Es handelt sich zudem um einen Variable-Length Code.
- Die Huffman-Codierung zählt zu den sogenannten Entropy-Encoding Verfahren, welche im Anschluss näher analysiert werden sollen.

Einheitliche Syntax

Der ITU Standard X.208 schlägt die folgende Methodik vor, um Daten mit unterschiedlichen syntaktischen Gegebenheiten zwischen zwei Systemen zu übertragen.



- Abstrakte Syntax beschreibt die Menge aller Darstellungstypen, die durch die Anwendungsprozesse definiert wurden.
- Die Kodierung dieser Darstellungstypen im lokalen System wird als lokale Syntax bezeichnet.
- Die abstrakte Syntax wird mittels einer sog. Transfersyntax zwischen den Instanzen über die Darstellungsschicht transferiert.
- Die Abbildung von einer abstrakten Syntax in eine konkrete Transfersyntax wird mittels Kodierregeln (engl. Encoding Rules) erreicht.
- Der Presentation Context bestimmt die Übergangsregeln für die De- und Enkodierung zwischen abstrakter Syntax und Transfersyntax.

Abstrakte Syntaxnotation Nummer 1: ASN.1

- ASN.1 ist eine abstrakte Syntax.
- Sie ist in der Backus-Naur-Notation gegeben und wird als Semantiksprache verwendet.

Definition (Backus-Naur-Form)

Die Backus-Naur-Form stellt eine Möglichkeit dar, die Syntax einer formalen Sprache zu beschreiben. In den BNF-Regeln sind folgende Elemente zugelassen:

- Syntaktische Variablen bzw. Nichtterminalsymbole (engl. nonterminals) dienen als Platzhalter für syntaktische Elemente. Nichtterminalsymbole werden mit <> beschrieben.
- Das Symbol **::=** dient einer Zuweisung.
- Terminalsymbole werden einzelnen Zeichen oder Zeichenketten der Wörter der Sprache bezeichnet.
- Operatoren ermöglichen die Verknüpfung von Terminalen und/oder Nichtterminalen.

Für die Verknüpfungen stehen folgenden Möglichkeiten zur Verfügung:

- Verkettung / Konkatenation
- Klammerung mehrere Ausdrücke: ( )
- Auswahl verschiedener Ausdrücke: |
- Wiederholungen einzelner Zeichen: *, um mehrere Zeichen zu wiederholen werden geschweifte Klammern { } verwendet
- optionale Ausdrücke: []

ASN.1 Beispiel

- Bei ASN.1 werden zusammengehörende Definitionen in Modulen gegliedert.
- Module können Definitionen exportieren (declarations) oder importieren (linkage)

```
ModulName DEFINITIONS ::= BEGIN
    linkage
    declarations
END
```

Definierbare Objekte in ASN.1 umfassen folgende Elemente:

- Typen (Types), um neue Datenstrukturen zu definieren, und werden mit Großbuchstaben am Beginn gekennzeichnet (`DatenTyp`)
- Werte (Values), um die Ausprägung der Typen zu definieren, werden lediglich in Kleinbuchstaben geschrieben (`zustand`)
- Makros (Macros), um die Grammatik der Sprache ändern zu können, werden wie alle anderen Schlüsselwörter in ASN.1 mit Großbuchstaben versehen (`OBJEKT`).

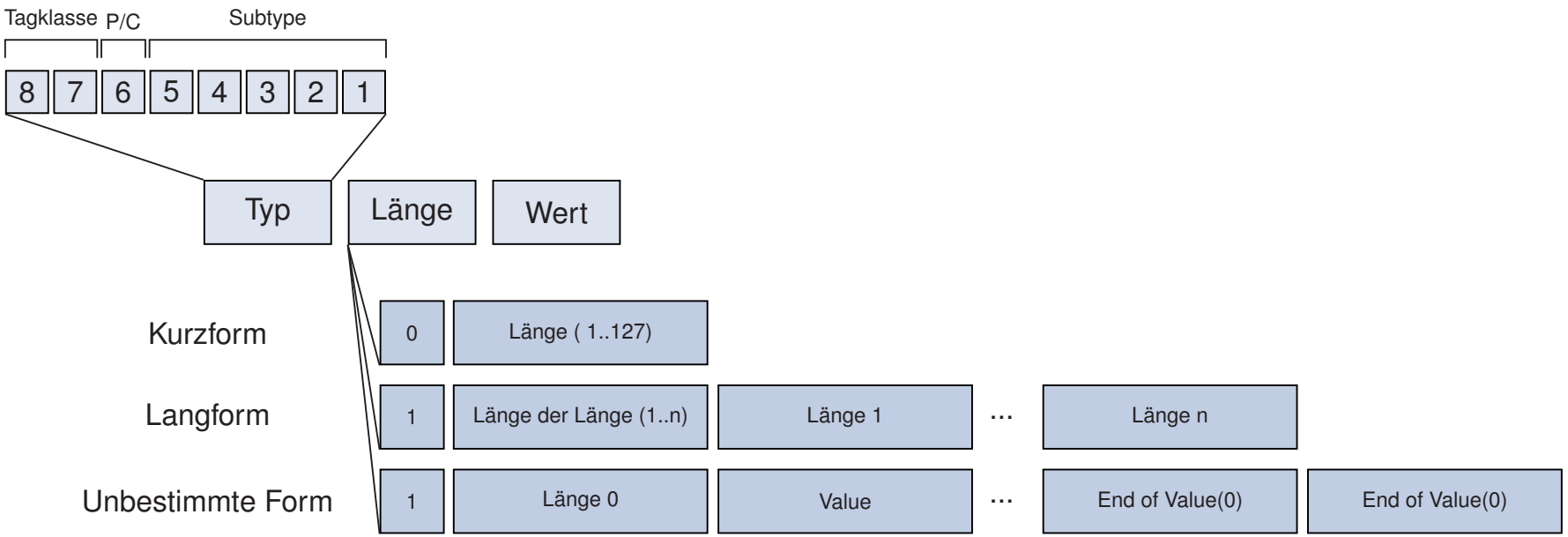
- Ein Typ wir durch `TypName ::= WERT` eingeführt, Variablen werden durch `VariablenName TypName ::= WERT` beschrieben.
- ASN.1 stellt eine Vielzahl von Datentypen bereit, die hier nicht näher beschrieben werden sollen.
- Eine genaue Beschreibung der Sprache findet sich in X.208 bzw. IS 8824 (für interessierte Studenten).

Basic Encoding Rules (BER)

Definition (Basic Encoding Rules, BER)

Die Basic Encoding Rules (BER) legen eine konkrete Transfersyntax fest, die definiert, wie die Datentypen der ASN.1 bei der Übertragung dargestellt werden. Sie sind in ISO 8825 definiert. Die Reihenfolge der Bits in einem Oktett und die Ordnung der Bytes selbst beginnt mit den Most Significant Bit (MSB) am Anfang.

In BER werden die Datentypen nach ASN.1 wie folgt kodiert und werden kurz mit TLV beschrieben:



Anmerkungen zur Kodierung in BER: Typ- oder Tag-Feld

Das Typ- bzw. Tag Feld enthält Informationen über die Klasse in den Bits 8 und 7, welches als class-Element bezeichnet wird. Mögliche Kodierungen sind

- Universal (**00**): ist reserviert für die Typen im ASN.1 Standard
- Application (**01**): ist gültig für eine Anwendung
- Context-Specific (**10**): ist gültig für eigene Spezifikationen
- Private (**11**): unterliegt keinen Einschränkungen, die Bedeutung geht aus dem Kontext hervor

Bit Nummer 6 wird als P/C-Bit bezeichnet und gibt an, ob es sich um einen primitiven oder konstrurierten Typ handelt.

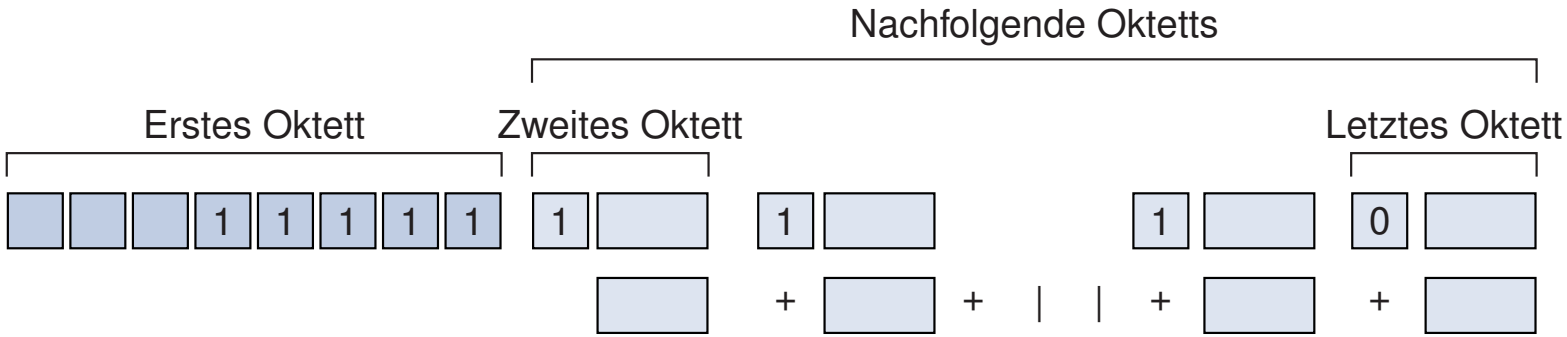
- **P/C = 1**: Der Value-Teil enthält wiederum eine BER-Struktur (TLV).
- **P/C = 0**: Der Value-Teil enthält nur ein einfaches Element

In den Bits 5 bis 1 wir der Subtype des Elements kodiert. Für die Klasse UNIVERSAL kann beispielsweise folgendes verwendet werden:

- **0**: End of Content (EOC)
- **1**: Boolean
- **2**: Integer
- Werte bis einschließlich 30 identifizieren weitere Subtypen, z.B. 4: OCTET STRING, 5: NULL
- **31**: Continued Number type: Die Identifikation des Subtypes befindet sich in einem späteren Feld

Anmerkungen zur Kodierung in BER: Continued Number Type

Der Continued Number Type hat folgende Struktur:



- Bit 8 jedes Oktetts wird auf 1 gesetzt um anzuzeigen, dass weitere Subtypefelder-Felder folgen.
- Wird das 8. Bit des Oktetts auf 0 gesetzt, ist das Ende des Subtypes erreicht.
- Diese Darstellung wird verwendet, wenn der Subtype durch einen größeren Wert als 30 identifiziert wird.

Anmerkungen zur Kodierung in BER: Längenfeld

Das Längenfeld gibt an, wie viele Datenoktetts folgen. Es können verschiedene Formen unterschieden werden:

- Kurzform: Das MSB (Most Significant Bit, d.h. Bit 8) wird auf 0 gesetzt. Die Bits 7 bis 1 codieren einen Wert zwischen 0 und 127, der die Länge des "ValueFeldes angibt..
- Langform: Das MSB (Most Significant Bit) wird auf 1 gesetzt und die Bits 7 bis 1 kodieren einen Wert zwischen 1 und 127, der die Anzahl nachfolgender Oktette des Längen-Felds angibt (Der Wert 127 ist reserviert für eventuelle Erweiterungen).
- Unbestimmte Form: Das MSB (Most Significant Bit) wird auf 1 gesetzt und die Bits 7 bis 1 auf 0 gesetzt. Es werden im "ValueFeld solange Oktetts übertragen bis der Delimiter (zwei auf Null gesetzte Oktette) verschickt wird. Es ist darauf zu achten, dass diese Struktur nicht in den zu versendenden Daten erscheint (s. Character Stuffing, Kapitel 4).

Anmerkungen zur Kodierung in BER: Längenfeld

Das Längenfeld gibt an, wie viele Datenoktetts folgen. Es können verschiedene Formen unterschieden werden:

- Kurzform: Das MSB (Most Significant Bit, d.h. Bit 8) wird auf 0 gesetzt. Die Bits 7 bis 1 codieren einen Wert zwischen 0 und 127, der die Länge des "ValueFeldes angibt..
- Langform: Das MSB (Most Significant Bit) wird auf 1 gesetzt und die Bits 7 bis 1 kodieren einen Wert zwischen 1 und 127, der die Anzahl nachfolgender Oktette des Längen-Felds angibt (Der Wert 127 ist reserviert für eventuelle Erweiterungen).
- Unbestimmte Form: Das MSB (Most Significant Bit) wird auf 1 gesetzt und die Bits 7 bis 1 auf 0 gesetzt. Es werden im "ValueFeld solange Oktetts übertragen bis der Delimiter (zwei auf Null gesetzte Oktette) verschickt wird. Es ist darauf zu achten, dass diese Struktur nicht in den zu versendenden Daten erscheint (s. Character Stuffing, Kapitel 4).

Frage: Wie viele Byte können mit Hilfe der jeweiligen Längenfelder übertragen werden?

Anmerkungen zur Kodierung in BER: Längenfeld

Das Längenfeld gibt an, wie viele Datenoktetts folgen. Es können verschiedene Formen unterschieden werden:

- Kurzform: Das MSB (Most Significant Bit, d.h. Bit 8) wird auf 0 gesetzt. Die Bits 7 bis 1 codieren einen Wert zwischen 0 und 127, der die Länge des "ValueFeldes angibt..
- Langform: Das MSB (Most Significant Bit) wird auf 1 gesetzt und die Bits 7 bis 1 kodieren einen Wert zwischen 1 und 127, der die Anzahl nachfolgender Oktette des Längen-Felds angibt (Der Wert 127 ist reserviert für eventuelle Erweiterungen).
- Unbestimmte Form: Das MSB (Most Significant Bit) wird auf 1 gesetzt und die Bits 7 bis 1 auf 0 gesetzt. Es werden im "ValueFeld solange Oktetts übertragen bis der Delimiter (zwei auf Null gesetzte Oktette) verschickt wird. Es ist darauf zu achten, dass diese Struktur nicht in den zu versendenden Daten erscheint (s. Character Stuffing, Kapitel 4).

Frage: Wie viele Byte können mit Hilfe der jeweiligen Längenfelder übertragen werden?

- Mit Hilfe der Kurzform: maximal 127 Byte. Die Langform erlaubt $126 * 8 = 1008$ Bit zur Kodierung der Länge: $2,47 * 10^{303}$ Byte

Anmerkungen zur Kodierung in BER: Längenfeld

Das Längenfeld gibt an, wie viele Datenoktetts folgen. Es können verschiedene Formen unterschieden werden:

- Kurzform: Das MSB (Most Significant Bit, d.h. Bit 8) wird auf 0 gesetzt. Die Bits 7 bis 1 codieren einen Wert zwischen 0 und 127, der die Länge des "ValueFeldes angibt..
- Langform: Das MSB (Most Significant Bit) wird auf 1 gesetzt und die Bits 7 bis 1 kodieren einen Wert zwischen 1 und 127, der die Anzahl nachfolgender Oktette des Längen-Felds angibt (Der Wert 127 ist reserviert für eventuelle Erweiterungen).
- Unbestimmte Form: Das MSB (Most Significant Bit) wird auf 1 gesetzt und die Bits 7 bis 1 auf 0 gesetzt. Es werden im "ValueFeld solange Oktetts übertragen bis der Delimiter (zwei auf Null gesetzte Oktette) verschickt wird. Es ist darauf zu achten, dass diese Struktur nicht in den zu versendenden Daten erscheint (s. Character Stuffing, Kapitel 4).

Frage: Wie viele Byte können mit Hilfe der jeweiligen Längenfelder übertragen werden?

- Mit Hilfe der Kurzform: maximal 127 Byte. Die Langform erlaubt $126 * 8 = 1008$ Bit zur Kodierung der Länge: $2,47 * 10^{303}$ Byte

Frage: Welchen Vorteil bringt eine unbestimmte Form, wenn die Anzahl der zu übertragenen Bytes der Langform ausreichen würde?

Anmerkungen zur Kodierung in BER: Längenfeld

Das Längenfeld gibt an, wie viele Datenoktetts folgen. Es können verschiedene Formen unterschieden werden:

- Kurzform: Das MSB (Most Significant Bit, d.h. Bit 8) wird auf 0 gesetzt. Die Bits 7 bis 1 codieren einen Wert zwischen 0 und 127, der die Länge des "ValueFeldes angibt..
- Langform: Das MSB (Most Significant Bit) wird auf 1 gesetzt und die Bits 7 bis 1 kodieren einen Wert zwischen 1 und 127, der die Anzahl nachfolgender Oktette des Längen-Felds angibt (Der Wert 127 ist reserviert für eventuelle Erweiterungen).
- Unbestimmte Form: Das MSB (Most Significant Bit) wird auf 1 gesetzt und die Bits 7 bis 1 auf 0 gesetzt. Es werden im "ValueFeld solange Oktetts übertragen bis der Delimiter (zwei auf Null gesetzte Oktette) verschickt wird. Es ist darauf zu achten, dass diese Struktur nicht in den zu versendenden Daten erscheint (s. Character Stuffing, Kapitel 4).

Frage: Wie viele Byte können mit Hilfe der jeweiligen Längenfelder übertragen werden?

- Mit Hilfe der Kurzform: maximal 127 Byte. Die Langform erlaubt $126 * 8 = 1008$ Bit zur Kodierung der Länge: $2,47 * 10^{303}$ Byte

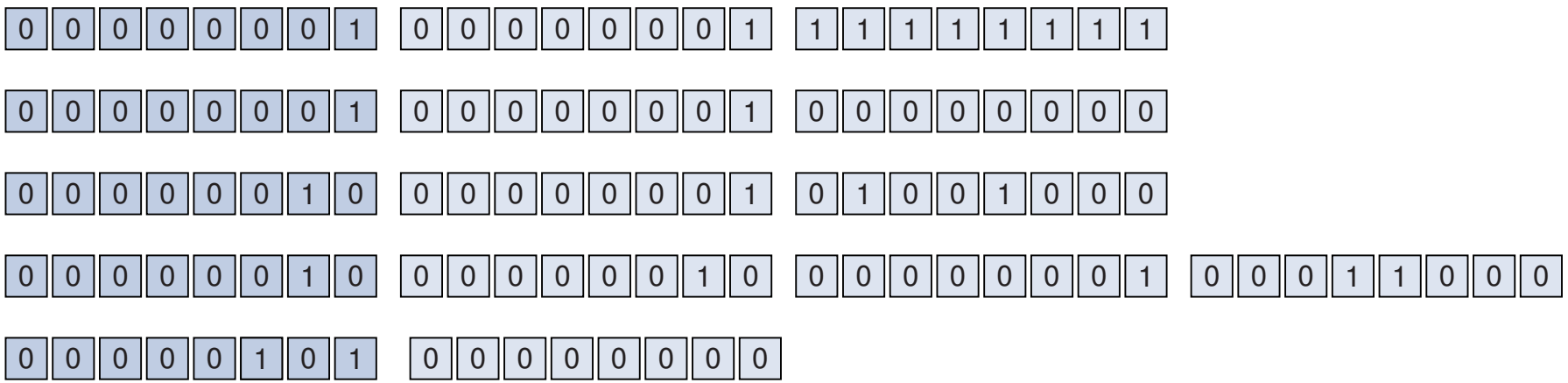
Frage: Welchen Vorteil bringt eine unbestimmte Form, wenn die Anzahl der zu übertragenen Bytes der Langform ausreichen würde?

- Die genaue Länge muss zu Beginn des Sendevorgangs nicht bekannt sein und die Übertragung kann beginnen, bevor die genaue Anzahl der Elemente bekannt ist.

BER: Beispiele

Es folgen einige Beispiele um die Basic Encoding Rules zu verdeutlichen:

- Boolean: TRUE und FALSE
- Integer: 72 und 280
- NULL



- Eine weitere konkrete Transfersyntax stellen die Packed Encoding Rules (PER) dar.
- Sie stellen die Nachfolgeregelung der BER dar und sind wesentlich effizienter vor allem in Bezug auf die Anzahl der übertragenen Bytes.
- Sie werden in den ITU-T X.691 und ISO 8825-2 spezifiziert.
- Praktischen Einsatz finden ASN.1 und die BER im Simple Network Management Protocol (SNMP).
- ASN.1 wird zur Beschreibung der SNMP-Pakete verwendet. Die Kodierung für den Transport wird mit Hilfe der BER erstellt.

Übersicht

Anwendungsschicht

Die Anwendungsschicht (Application Layer) stellt Anwendungen spezifische Dienste sowie Zugang zu Diensten darunter liegender Schichten zur Verfügung. Protokolle der Anwendungsschicht können Teil einer Anwendung sein (z. B. bei Webserver oder Webbrowser).

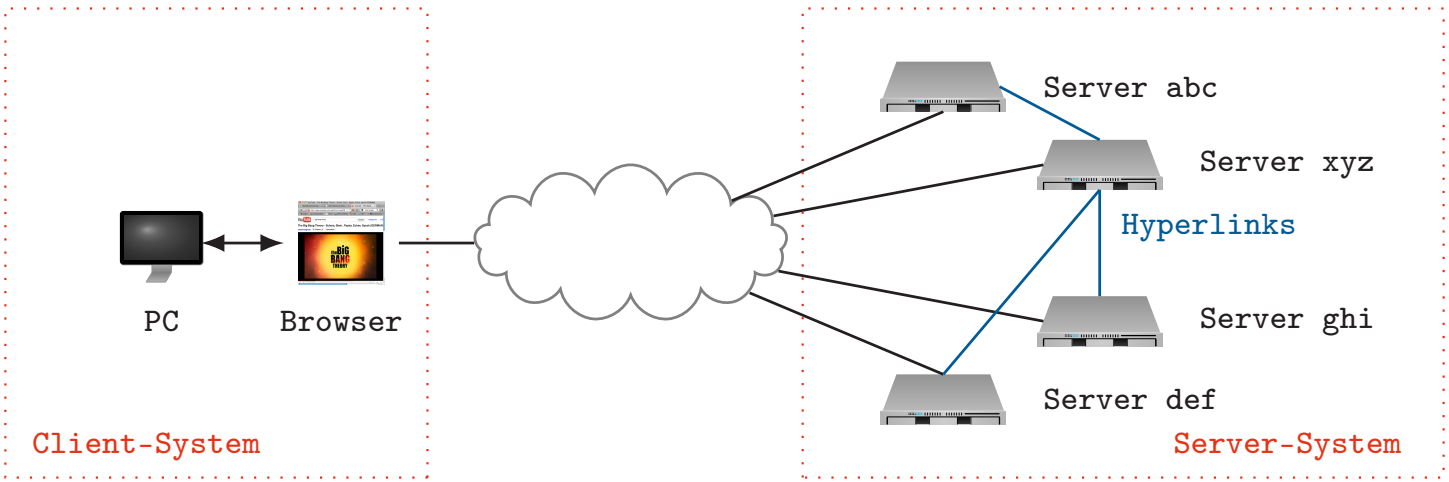
Beispiele für Dienste der Anwendungsschicht:

- Namensauflösung (DNS)
- Dateitransfer (HTTP, FTP)
- Nachrichtentransfer (SMTP, POP, IMAP, NNTP)
- Entfernte Anmeldung (Telnet, SSH)
- Netzmanagement (SNMP)

Im Folgenden behandeln wir DNS, HTTP und SMTP als wichtige Protokolle der Anwendungsschicht etwas genauer.

World Wide Web (WWW)

- Um die Protokolle DNS und HTTP besser einordnen zu können, werfen wir einen kurzen Blick auf das World Wide Web, kurz Web.
- Es handelt sich dabei um das architektonische Rahmenwerk zum Dokumentzugriff im Internet.
- 1994 wurde das World Wide Web Consortium (W3C) mit dem Ziel, Web-Technologien weiterzuentwickeln und zu standardisieren.
- Das WWW begründet sich auf einer Client-Server-Architektur, die im Folgenden kurz dargestellt wird.



- Zur client-seitigen Darstellung von Web-Seiten (Pages) wird ein Browser verwendet.
- Hyperlinks verknüpfen unterschiedliche Seiten miteinander.

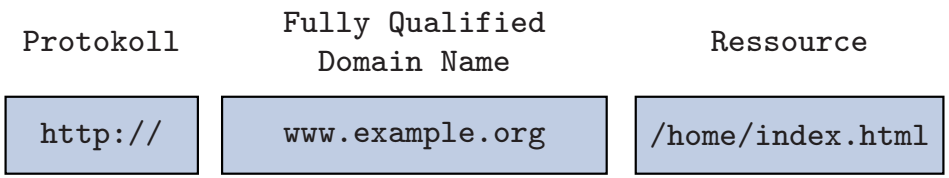
Typischer Ablauf aus Sicht des Clients beim Aufruf einer Web-Seite

- Browser prüft bzw. parst URL (Uniform Resource Locator)

Typischer Ablauf aus Sicht des Clients beim Aufruf einer Web-Seite

- Browser prüft bzw. parst URL (Uniform Resource Locator)

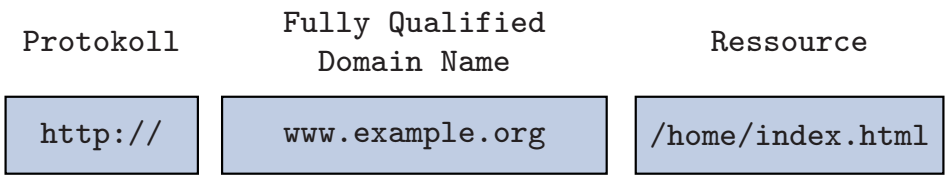
Eine URL besteht im Wesentlichen aus drei Bestandteilen:



Typischer Ablauf aus Sicht des Clients beim Aufruf einer Web-Seite

- Browser prüft bzw. parst URL (Uniform Resource Locator)

Eine URL besteht im Wesentlichen aus drei Bestandteilen:

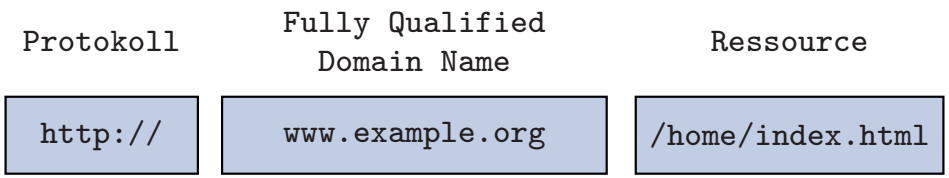


- Ermittlung der IP-Adresse mit Hilfe von DNS

Typischer Ablauf aus Sicht des Clients beim Aufruf einer Web-Seite

- Browser prüft bzw. parst URL (Uniform Resource Locator)

Eine URL besteht im Wesentlichen aus drei Bestandteilen:

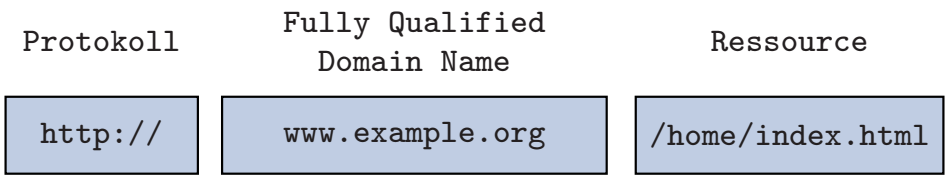


- Ermittlung der IP-Adresse mit Hilfe von DNS
- Aufbau einer Verbindung (TCP) über Port 80 zur ermittelten IP-Adresse

Typischer Ablauf aus Sicht des Clients beim Aufruf einer Web-Seite

- Browser prüft bzw. parst URL (Uniform Resource Locator)

Eine URL besteht im Wesentlichen aus drei Bestandteilen:

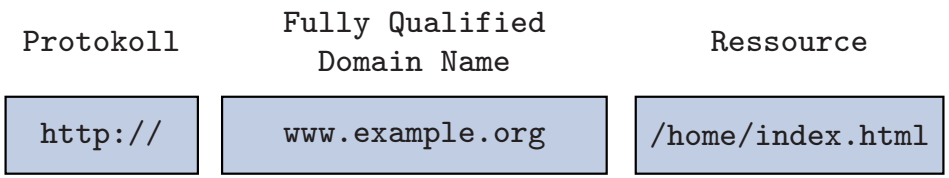


- Ermittlung der IP-Adresse mit Hilfe von DNS
- Aufbau einer Verbindung (TCP) über Port 80 zur ermittelten IP-Adresse
- Anfordern der Datei /home/index.html

Typischer Ablauf aus Sicht des Clients beim Aufruf einer Web-Seite

- Browser prüft bzw. parst URL (Uniform Resource Locator)

Eine URL besteht im Wesentlichen aus drei Bestandteilen:

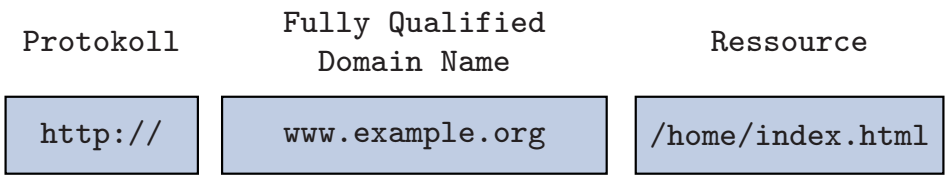


- Ermittlung der IP-Adresse mit Hilfe von DNS
- Aufbau einer Verbindung (TCP) über Port 80 zur ermittelten IP-Adresse
- Anfordern der Datei /home/index.html

Typischer Ablauf aus Sicht des Clients beim Aufruf einer Web-Seite

- Browser prüft bzw. parst URL (Uniform Resource Locator)

Eine URL besteht im Wesentlichen aus drei Bestandteilen:

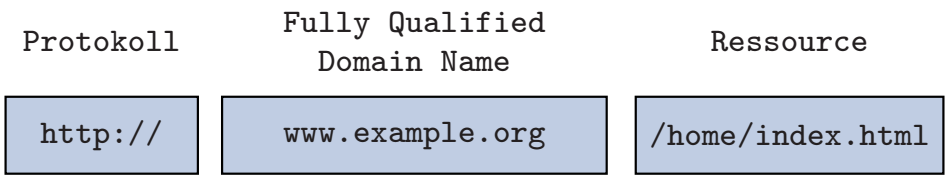


- Ermittlung der IP-Adresse mit Hilfe von DNS
- Aufbau einer Verbindung (TCP) über Port 80 zur ermittelten IP-Adresse
- Anfordern der Datei /home/index.html

Typischer Ablauf aus Sicht des Clients beim Aufruf einer Web-Seite

- Browser prüft bzw. parst URL (Uniform Resource Locator)

Eine URL besteht im Wesentlichen aus drei Bestandteilen:

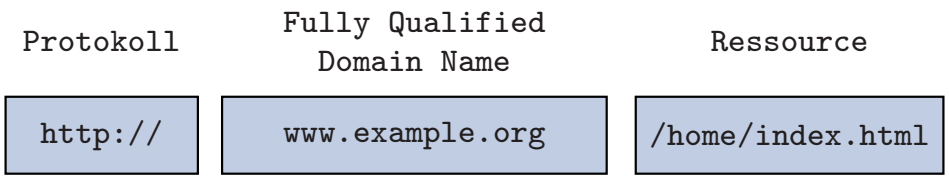


- Ermittlung der IP-Adresse mit Hilfe von DNS
- Aufbau einer Verbindung (TCP) über Port 80 zur ermittelten IP-Adresse
- Anfordern der Datei /home/index.html

Typischer Ablauf aus Sicht des Clients beim Aufruf einer Web-Seite

- Browser prüft bzw. parst URL (Uniform Resource Locator)

Eine URL besteht im Wesentlichen aus drei Bestandteilen:

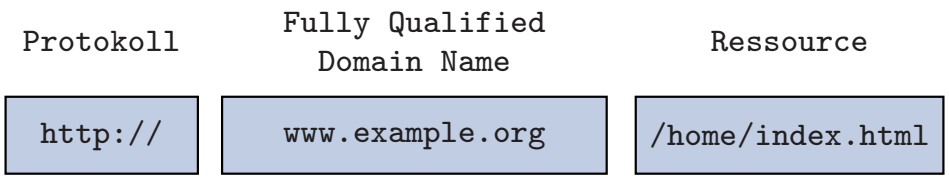


- Ermittlung der IP-Adresse mit Hilfe von DNS
- Aufbau einer Verbindung (TCP) über Port 80 zur ermittelten IP-Adresse
- Anfordern der Datei /home/index.html

Typischer Ablauf aus Sicht des Clients beim Aufruf einer Web-Seite

- Browser prüft bzw. parst URL (Uniform Resource Locator)

Eine URL besteht im Wesentlichen aus drei Bestandteilen:



- Ermittlung der IP-Adresse mit Hilfe von DNS
- Aufbau einer Verbindung (TCP) über Port 80 zur ermittelten IP-Adresse
- Anfordern der Datei /home/index.html

Domain Name System (DNS)

Motivation:

- Möchte ein Nutzer (Mensch) einen Computer adressieren, will er sich gewöhnlich nicht dessen IP-Adresse merken müssen.
- Stattdessen adressiert man das Ziel i. d. R. mittles eines hierarchisch aufgebauten Namens, z. B. www.google.com.

Domain Name System (DNS)

Motivation:

- Möchte ein Nutzer (Mensch) einen Computer adressieren, will er sich gewöhnlich nicht dessen IP-Adresse merken müssen.
- Stattdessen adressiert man das Ziel i. d. R. mittles eines hierarchisch aufgebauten Namens, z. B. www.google.com.

Domain Name System (DNS):

- Ein DNS-Name (Fully Qualified Domain Name, FQDN) besteht aus einem Hostnamen und einem Suffix:

Domain Name System (DNS)

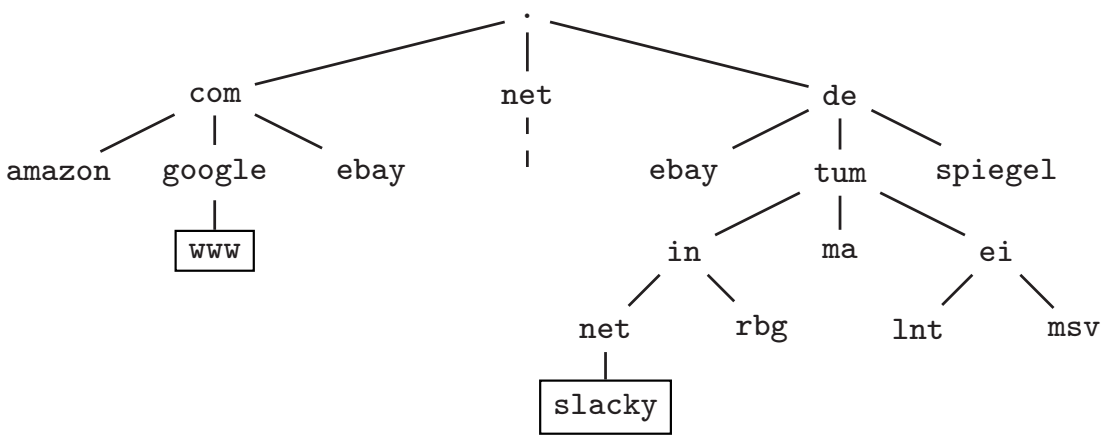
Motivation:

- Möchte ein Nutzer (Mensch) einen Computer adressieren, will er sich gewöhnlich nicht dessen IP-Adresse merken müssen.
- Stattdessen adressiert man das Ziel i. d. R. mittles eines hierarchisch aufgebauten Namens, z. B. www.google.com.

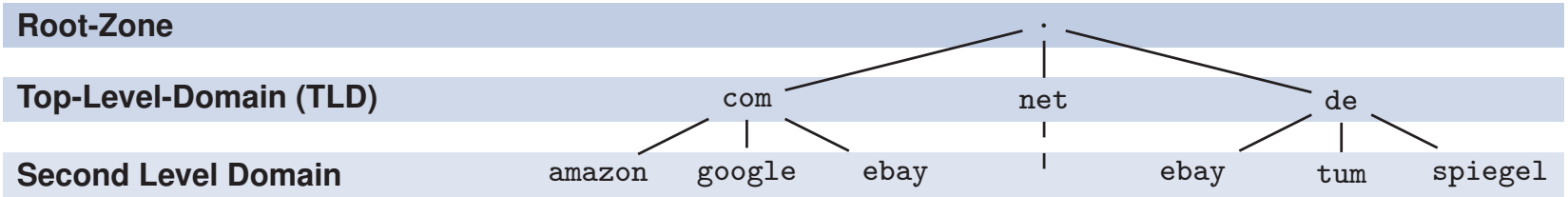
Domain Name System (DNS):

- Ein DNS-Name (Fully Qualified Domain Name, FQDN) besteht aus einem Hostnamen und einem Suffix:

- Das Suffix ist hierarchisch von rechts nach links beginnend bei Root (.) aufgebaut:



Wichtige Begriffe



- Der Namensraum ist in Zonen unterteilt, die unabhängig voneinander administriert werden.
- In jeder Zone gibt es einen primären DNS-Server, welcher
  - eine Liste aller Hosts in dieser Zone und
  - eine Liste mit autoritativen DNS-Servern untergeordneter Domänen

  verwaltet. Änderungen an den Einträgen geschehen nur über den autoritativen DNS-Server.
- Zusätzlich kann es eine Reihe sekundärer Server geben, welche eine Kopie der Zonendatei besitzen ("Zone Transfer", für Lastverteilung und Ausfallsicherheit).
- Die Antworten dieser Server werden als autoritativ bezeichnet, da die Antwort auf Konfigurationsinformation beruht und nicht per DNS von einem anderen Server erhalten wurde. Autoritative Antworten werden von anderen DNS-Servern gecacht.
- Die Cachedauer hängt vom TTL-Wert ab, welcher in jeder Antwort eines Servers enthalten ist.
- Gibt ein DNS-Server einen derartigen gecachten Eintrag weiter, so spricht man von einer nicht-autoritativen Antwort.
- Infolge des Cachings kann es mehrere Stunden (u. U. auch Tage) dauern, bis Änderungen am primären DNS-Server einer Zone im Internet vollständig sichtbar werden.
- Weitere Details zu DNS finden sich u. a. in RFC1035.

In der Zonendatei finden sich DNS Resource Records unterschiedlicher Typen, u. a.:

- SOA – Start Of Authority Enthält Angaben zur Verwaltung der Zone, u. a.
  - Zonenklasse (IN für Internet, HS für Hesiod) Der Hesiod Name Service stammt aus dem Athena-Projekt. Mit ihm lassen sich Informationen aus /etc/passwd etc. zugänglich machen. Eine heute übliche Methode besteht darin, diese Information per LDAP zugänglich zu machen.
  - Seriennummer
  - Refresh-Abstand (in Sekunden) für Anfragen nach Änderungen
  - Retry-Abstand zur Wiederholung nicht beantworteter Anfragen
  - Expire-Zeit nach der eine Zone von einem Slave nach einem Zonentransfer-Request als deaktiviert betrachtet wird
  - negativ-Caching-TTL, wenn zu einem angefragten Domainnamen kein Eintrag zugeordnet ist
- A – Hosteintrag IPv4
  Enthält das Mapping zwischen einem Hostnamen und einer IPv4-Adresse
- AAAA – Hosteintrag IPv6
  Enthält das Mapping zwischen einem Hostnamen und einer IPv6-Adresse
- NS – Name Server
  Gibt den FQDN eines DNS-Servers an, welcher für eine (untergeordnete) Domäne autoritativ ist
- CNAME – Common Name
  Ermöglicht es, einem bestehenden Hosteintrag einen weiteren Namen zuzuordnen, z. B. könnte für den Host `typo3.net.in.tum.de.` ein CNAME-Eintrag vorhanden sein, so dass dieser auch unter dem Namen `www.net.in.tum.de.` erreichbar wird.
- MX – Mail Exchanger
  Gibt einen Mailserver (SMTP-Server) für eine Domäne an.

Allgemeine Anmerkungen:

- Einer IP-Adresse können im Prinzip beliebig viele Namen zugewiesen werden.
- Umgekehrt können sich hinter einem Namen auch mehrere Adressen verbergen (einfach mal `nslookup google.com` auf der Kommandozeile ausführen). Warum könnte das sinnvoll sein?
- DNS-Server können auch Reverse-Lookups durchführen, sofern sie dazu konfiguriert wurden (Übersetzung einer IP-Adresse in den/die zugehörigen DNS-Name(n)).

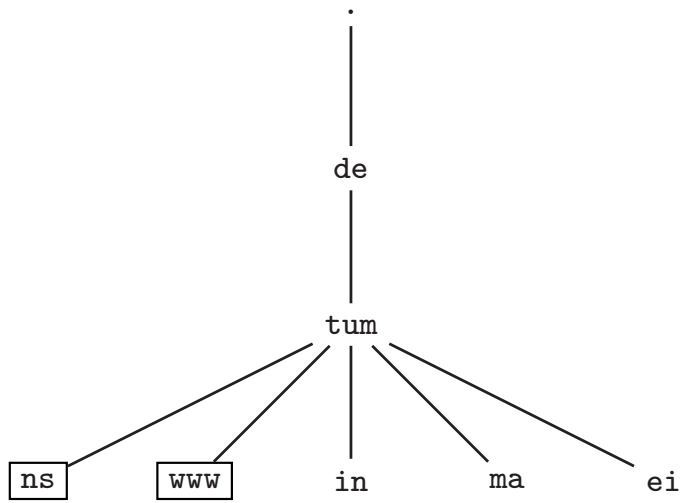
Beispiel: Zonendatei des DNS-Servers `ns.tum.de`

```
$TTL 1W
@  IN  SOA  ns.tum.de. mail.tum.de. (
         2007100801 ; serial
         28800 ; refresh (8 Stunden)
         7200 ; retry (2 Stunden)
         604800 ; expire (1 Woche)
         39600 ; minimum (11 Stunden)
         )

tum.de.         IN  NS  ns.tum.de.
in.tum.de.      IN  NS  ns.in.tum.de.
ma.tum.de.      IN  NS  ns.ma.tum.de.
ei.tum.de.      IN  NS  ns.ei.tum.de.

ns.in.tum.de.   IN  A  <IP von ns.in.tum.de.>
ns.ma.tum.de.   IN  A  <IP von ns.ma.tum.de.>
ns.ei.tum.de.   IN  A  <IP von ns.ei.tum.de.>
ns              IN  A  <IP von ns.tum.de.>
www             IN  A  <IP von www.tum.de.>
```

Ausschnitt aus dem DNS-Namensraum:



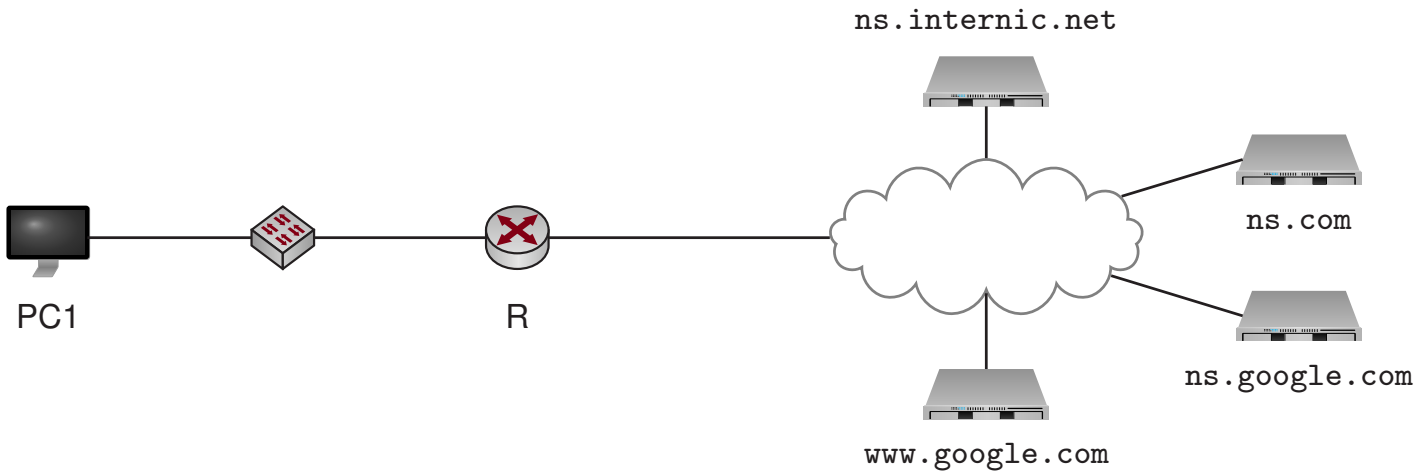
Die Zonendatei von `ns.tum.de.` beinhaltet

- allgemeine Angaben zur Zonendatei (Gültigkeitsdauer, Seriennummer, Refresh-Zeiten usw.),
- eine Liste der autoritativen Namensserver für `tum.de.` selbst sowie alle untergeordneten Domänen, für die `ns.tum.de.` nicht selbst zuständig ist und
- eine Liste mit allen Hostnamen der Domäne(n), für die `ns.tum.de.` selbst zuständig ist.

Wie funktioniert das jetzt aber genau?

Wir nehmen an, PC1 möchte die zum DNS-Namen `www.google.com` gehörende IP-Adresse auflösen. Übliches Setup:

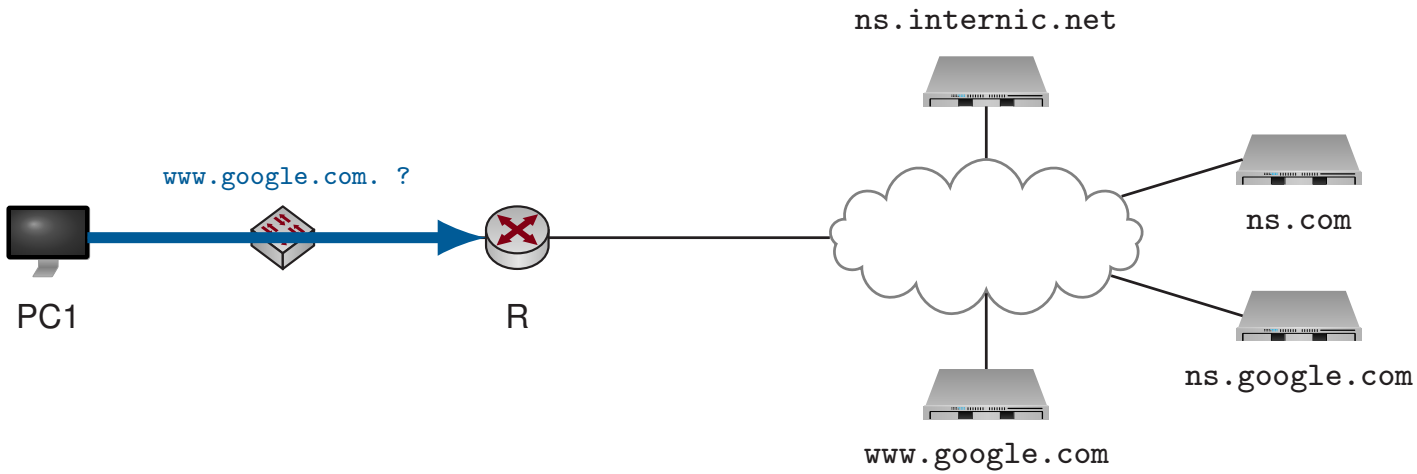
- PC1 ist mit der lokalen IP-Adresse seines Routers R als DNS-Server konfiguriert.
- R selbst ist für keine Zone autoritativ. Seine einzige Aufgabe besteht darin, für Hosts im lokalen Netzwerk die Namensauflösung zu übernehmen.



Wie funktioniert das jetzt aber genau?

Wir nehmen an, PC1 möchte die zum DNS-Namen `www.google.com` gehörende IP-Adresse auflösen. Übliches Setup:

- PC1 ist mit der lokalen IP-Adresse seines Routers R als DNS-Server konfiguriert.
- R selbst ist für keine Zone autoritativ. Seine einzige Aufgabe besteht darin, für Hosts im lokalen Netzwerk die Namensauflösung zu übernehmen.

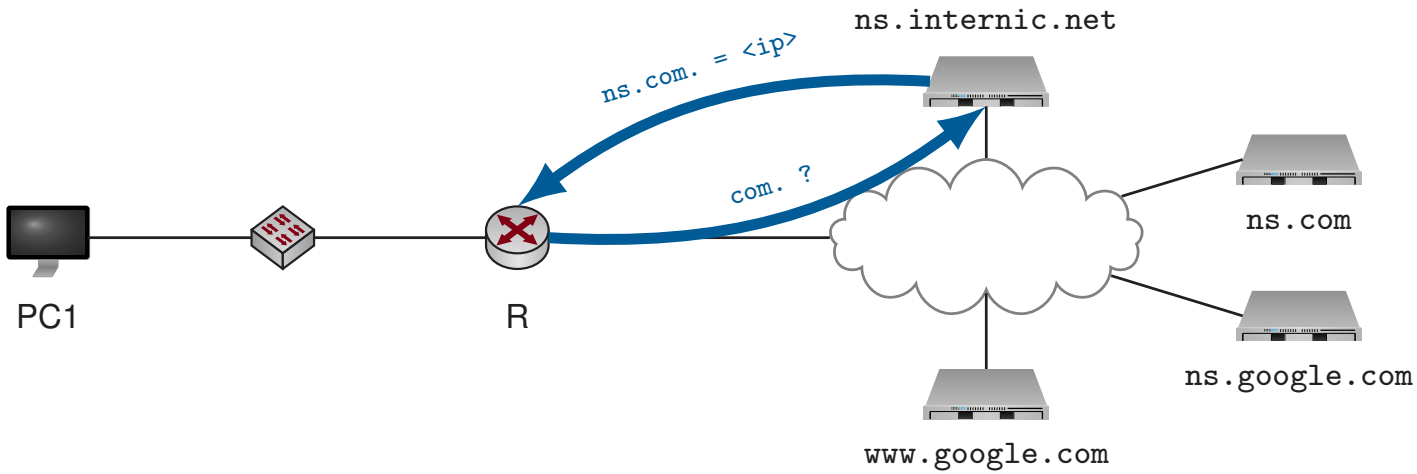


- PC1 schickt an R einen DNS-Request, um `www.google.com` aufzulösen.

Wie funktioniert das jetzt aber genau?

Wir nehmen an, PC1 möchte die zum DNS-Namen `www.google.com` gehörende IP-Adresse auflösen. Übliches Setup:

- PC1 ist mit der lokalen IP-Adresse seines Routers R als DNS-Server konfiguriert.
- R selbst ist für keine Zone autoritativ. Seine einzige Aufgabe besteht darin, für Hosts im lokalen Netzwerk die Namensauflösung zu übernehmen.

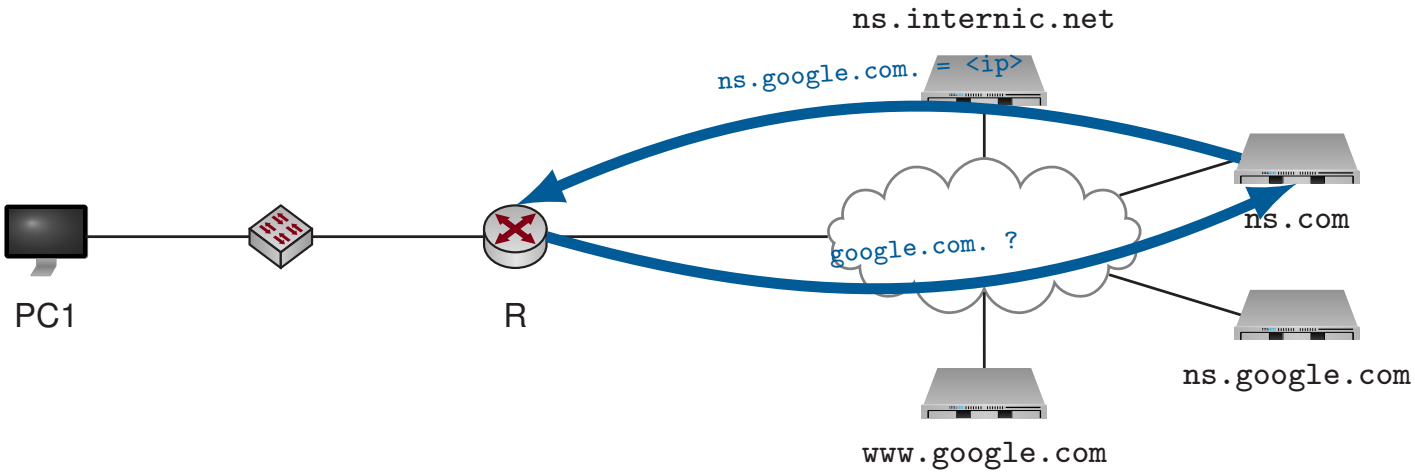


- PC1 schickt an R einen DNS-Request, um `www.google.com` aufzulösen.
- R kennt weder `www.google.com`, noch `google.com` oder `com` selbst und fragt deshalb bei einem der Root-Server an, welche DNS-Server für die Zone `com.` verantwortlich sind.

Wie funktioniert das jetzt aber genau?

Wir nehmen an, PC1 möchte die zum DNS-Namen `www.google.com` gehörende IP-Adresse auflösen. Übliches Setup:

- PC1 ist mit der lokalen IP-Adresse seines Routers R als DNS-Server konfiguriert.
- R selbst ist für keine Zone autoritativ. Seine einzige Aufgabe besteht darin, für Hosts im lokalen Netzwerk die Namensauflösung zu übernehmen.

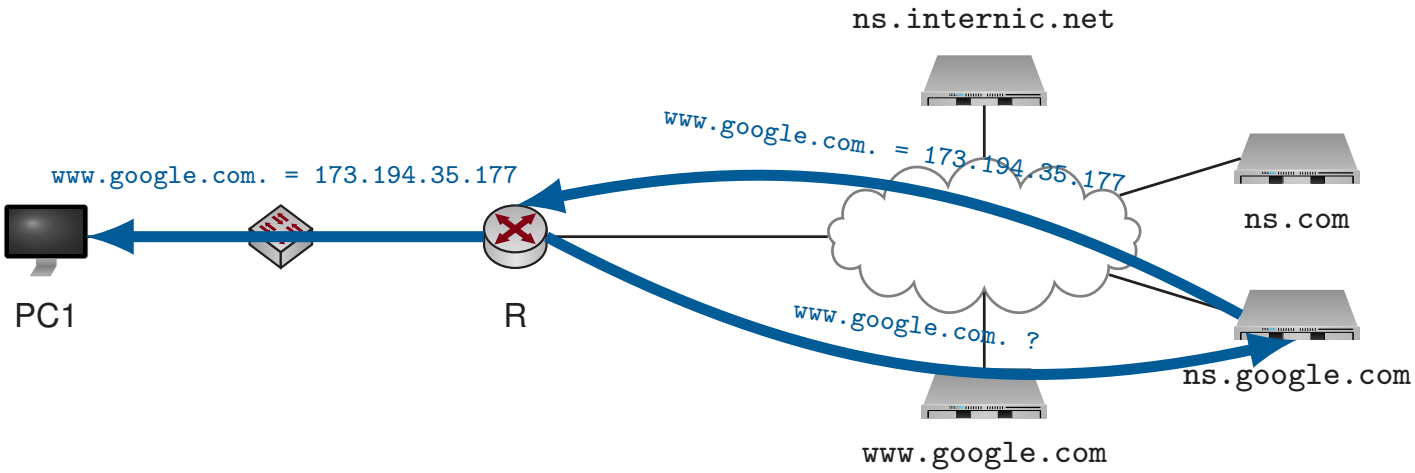


- PC1 schickt an R einen DNS-Request, um `www.google.com` aufzulösen.
- R kennt weder `www.google.com`, noch `google.com` oder `com` selbst und fragt deshalb bei einem der Root-Server an, welche DNS-Server für die Zone `com.` verantwortlich sind.
- Anschließend fragt R bei `ns.com`, wer für `google.com` verantwortlich ist.

Wie funktioniert das jetzt aber genau?

Wir nehmen an, PC1 möchte die zum DNS-Namen `www.google.com` gehörende IP-Adresse auflösen. Übliches Setup:

- PC1 ist mit der lokalen IP-Adresse seines Routers R als DNS-Server konfiguriert.
- R selbst ist für keine Zone autoritativ. Seine einzige Aufgabe besteht darin, für Hosts im lokalen Netzwerk die Namensauflösung zu übernehmen.



- PC1 schickt an R einen DNS-Request, um `www.google.com` aufzulösen.
- R kennt weder `www.google.com`, noch `google.com` oder `com` selbst und fragt deshalb bei einem der Root-Server an, welche DNS-Server für die Zone `com.` verantwortlich sind.
- Anschließend fragt R bei `ns.com`, wer für `google.com` verantwortlich ist.
- Schließlich wendet sich R an `ns.google.com` um die IP-Adresse von `www.google.com` aufzulösen. Das Ergebnis wird an PC1 weitergeleitet und im Cache von R gespeichert (TTL ist in der Antwort von `ns.google.com` enthalten).

Rekursive Namensauflösung

- PC1 sendet einmal einen Request an seinen DNS-Server und erwartet als Response das Ergebnis der Namensauflösung.
- Welche Schritte R unternehmen muss (oder unternimmt), um den Request zu beantworten, weiß PC1 nicht.
- Zwischen PC1 und R werden also nur zwei Pakete ausgetauscht.

Iterative Namensauflösung

- Sofern R nicht bereits Teile des angefragten Namens in seinem Cache hat (z. B. den ganzen FQDN oder einen autoritativen DNS-Server für `google.com`), wird der FQDN Schritt für Schritt (iterativ) beginnend bei der Wurzel aufgelöst.
- Die IP-Adressen der DNS-Rootserver hat jeder DNS-Server (also auch der minimalistische Server auf R) gespeichert. Diese bezeichnet man als Root Hints.
- Prinzipiell wäre es natürlich auch möglich (und ist eher die Regel als die Ausnahme), dass R selbst zur rekursiven Namensauflösung konfiguriert ist. In diesem Fall würde R den Request von PC1 einfach an seinen DNS-Server weiterleiten, z. B. den DNS-Server eines Service Providers wie der Deutschen Telekom.

HyperText Transfer Protocol (HTTP)

- Als Standardübertragungsprotokoll im Web wird definiert, welche Anfragen ein Client senden und wie der Server antworten darf.
- Es handelt sich um ein ASCII-Anwendungsprotokoll, welches im RFC 2616 in der Version HTTP 1.1 definiert ist.
- Es setzt auf eine TCP-Verbindung auf und verwendet standardmäßig den Port 80.
- In der Version HTTP 1.0 wird vom Server nach jeder beantworteten Anfrage die TCP-Verbindung abgebaut.

HyperText Transfer Protocol (HTTP)

- Als Standardübertragungsprotokoll im Web wird definiert, welche Anfragen ein Client senden und wie der Server antworten darf.
- Es handelt sich um ein ASCII-Anwendungsprotokoll, welches im RFC 2616 in der Version HTTP 1.1 definiert ist.
- Es setzt auf eine TCP-Verbindung auf und verwendet standardmäßig den Port 80.
- In der Version HTTP 1.0 wird vom Server nach jeder beantworteten Anfrage die TCP-Verbindung abgebaut.

Welche Nachteile ergeben sich, wenn eine Web-Seite mit vielen Grafiken angefragt wird?

HyperText Transfer Protocol (HTTP)

- Als Standardübertragungsprotokoll im Web wird definiert, welche Anfragen ein Client senden und wie der Server antworten darf.
- Es handelt sich um ein ASCII-Anwendungsprotokoll, welches im RFC 2616 in der Version HTTP 1.1 definiert ist.
- Es setzt auf eine TCP-Verbindung auf und verwendet standardmäßig den Port 80.
- In der Version HTTP 1.0 wird vom Server nach jeder beantworteten Anfrage die TCP-Verbindung abgebaut.

Welche Nachteile ergeben sich, wenn eine Web-Seite mit vielen Grafiken angefragt wird?

- Jede einzelne Grafik wird mit einer eigenen TCP-Verbindung übertragen.

HyperText Transfer Protocol (HTTP)

- Als Standardübertragungsprotokoll im Web wird definiert, welche Anfragen ein Client senden und wie der Server antworten darf.
- Es handelt sich um ein ASCII-Anwendungsprotokoll, welches im RFC 2616 in der Version HTTP 1.1 definiert ist.
- Es setzt auf eine TCP-Verbindung auf und verwendet standardmäßig den Port 80.
- In der Version HTTP 1.0 wird vom Server nach jeder beantworteten Anfrage die TCP-Verbindung abgebaut.

Welche Nachteile ergeben sich, wenn eine Web-Seite mit vielen Grafiken angefragt wird?

- Jede einzelne Grafik wird mit einer eigenen TCP-Verbindung übertragen.
- **Verbesserung:** HTTP 1.1 verwendet sog. persistent connections (mit typischem timeout 5-15 sec).

HyperText Transfer Protocol (HTTP)

- Als Standardübertragungsprotokoll im Web wird definiert, welche Anfragen ein Client senden und wie der Server antworten darf.
- Es handelt sich um ein ASCII-Anwendungsprotokoll, welches im RFC 2616 in der Version HTTP 1.1 definiert ist.
- Es setzt auf eine TCP-Verbindung auf und verwendet standardmäßig den Port 80.
- In der Version HTTP 1.0 wird vom Server nach jeder beantworteten Anfrage die TCP-Verbindung abgebaut.

Welche Nachteile ergeben sich, wenn eine Web-Seite mit vielen Grafiken angefragt wird?

- Jede einzelne Grafik wird mit einer eigenen TCP-Verbindung übertragen.
- **Verbesserung:** HTTP 1.1 verwendet sog. persistent connections (mit typischem timeout 5-15 sec).

- Im Folgenden werden HTTP-Requests und HTTP-Responses betrachtet.
- Anhand eines Beispiels wird der Protokollablauf verdeutlicht.

HTTP - Request

- Eine Anfrage des Clients an den Server wird als HTTP-Request bezeichnet.
- Ein HTTP-Request besteht aus der verwendeten Methode und der betreffenden URL.

Bezeichnung	Beschreibung
GET	Der Client fordert den Server auf, eine bestimmte Ressource zu versenden
HEAD	Der Client fordert den Server auf, den Header eine bestimmte Ressource zu versenden
PUT	Das Schreiben einer Ressource wird angefordert
POST	Es werden neue Daten an die benannte Ressource angehängt
DELETE	Es wird angefordert eine Ressource zu löschen

- Ein Request-Header kann zusätzliche Informationen enthalten.

Header	Beschreibung
User-Agent	Browser- und Plattforminformationen
Accept	durch den Client unterstützte MIME-Typen, z. B. *text/html*
Accept-Charset	durch den Client unterstützte Zeichensätze, z. B. *Unicode*
Accept-Encoding	durch den Client unterstützte Kompressionsverfahren, z. B. *gzip*
Accept-Language	natürliche Sprache, z. B. *english*
Host	DNS-Name des Servers
Authorization	Anmeldeinformationen des Clients
Cookies	vom Server erhaltene Cookies

HTTP - Response

- Ein HTTP-Response beginnt mit der Statuszeile mit einem entsprechenden Status-Code.

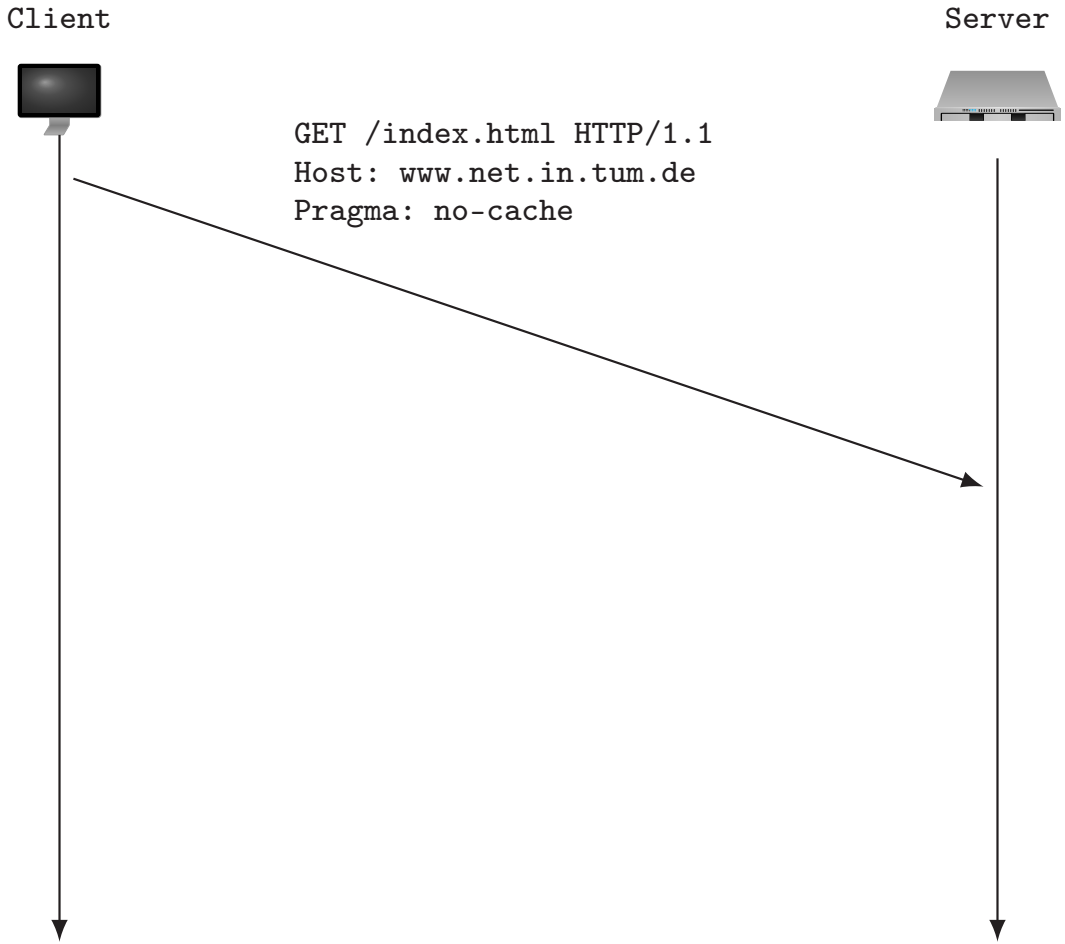
Code	Bedeutung	Beispiel
1xx	Information	102 = *Processing*, zur Verhinderung eines Timeouts
2xx	Erfolg	200 = *OK*, die Anfrage wurde erfolgreich bearbeitet
3xx	Umleitung	301 = *Moved Permanently*, die alte Adresse ist nicht länger gültig
4xx	Client-Fehler	400 = *Bad Request*, fehlerhaft aufgabeute Anfrage Nachricht
5xx	Server-Fehler	503 = *Service Unavailable*, Server steht temporär nicht zur Verfügung

- Nach der Statuszeile folgt der Response-Header, sowie ggf. die angeforderte Ressource.

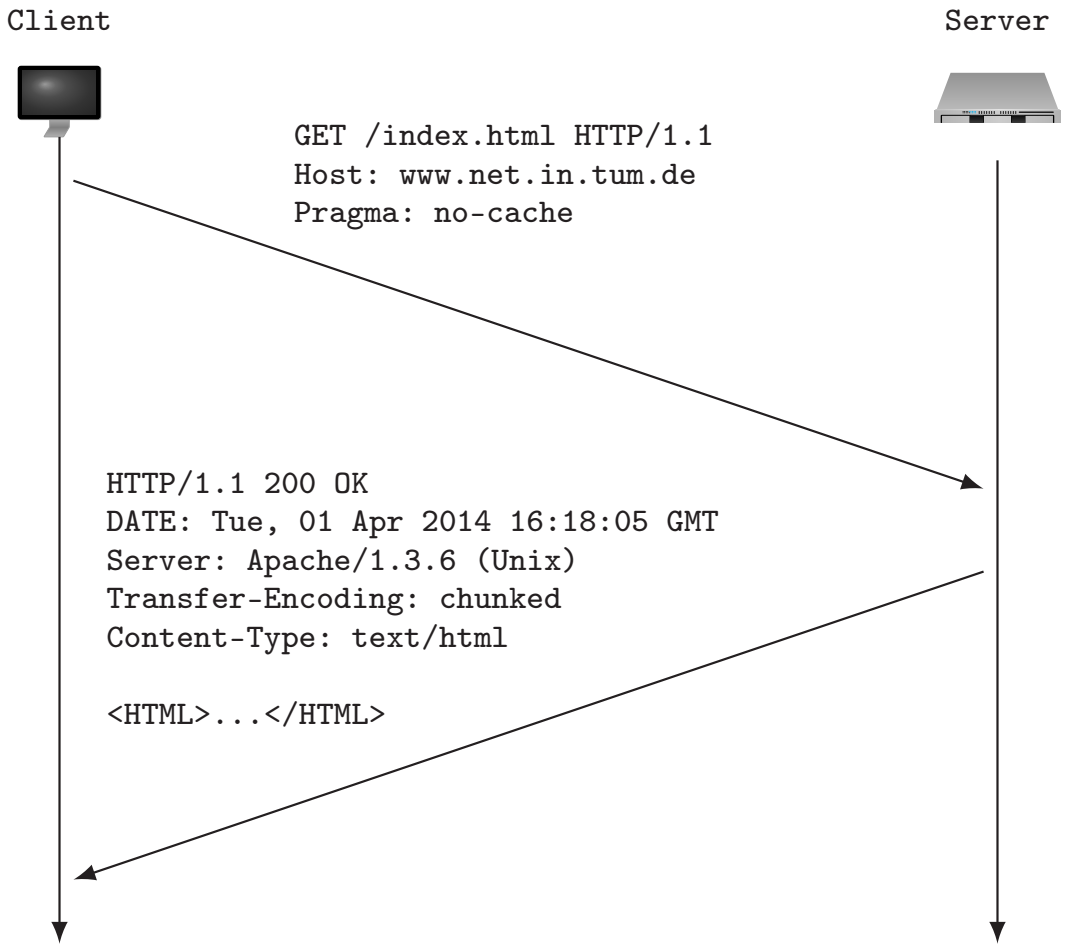
Response-Header	Beschreibung
Server	Informationen über den Server
Content-Lenght	Seitenlänge in Byte
Content-Encoding	Kodierungsinformationen
Content-Language	natürliche Sprache der Seite
Content-Type	MIME-Type der Seite
Last-Modified	Letztes Änderungsdatum der Seite
Location	Anforderungen müssen an gegeben Server gesendet werden
Accept-Ranges	Server akzeptiert Bereichsanfragen in Byte
Set-Cookies	Aufforderung an den Client ein Cookie zu speichern

Beispiel

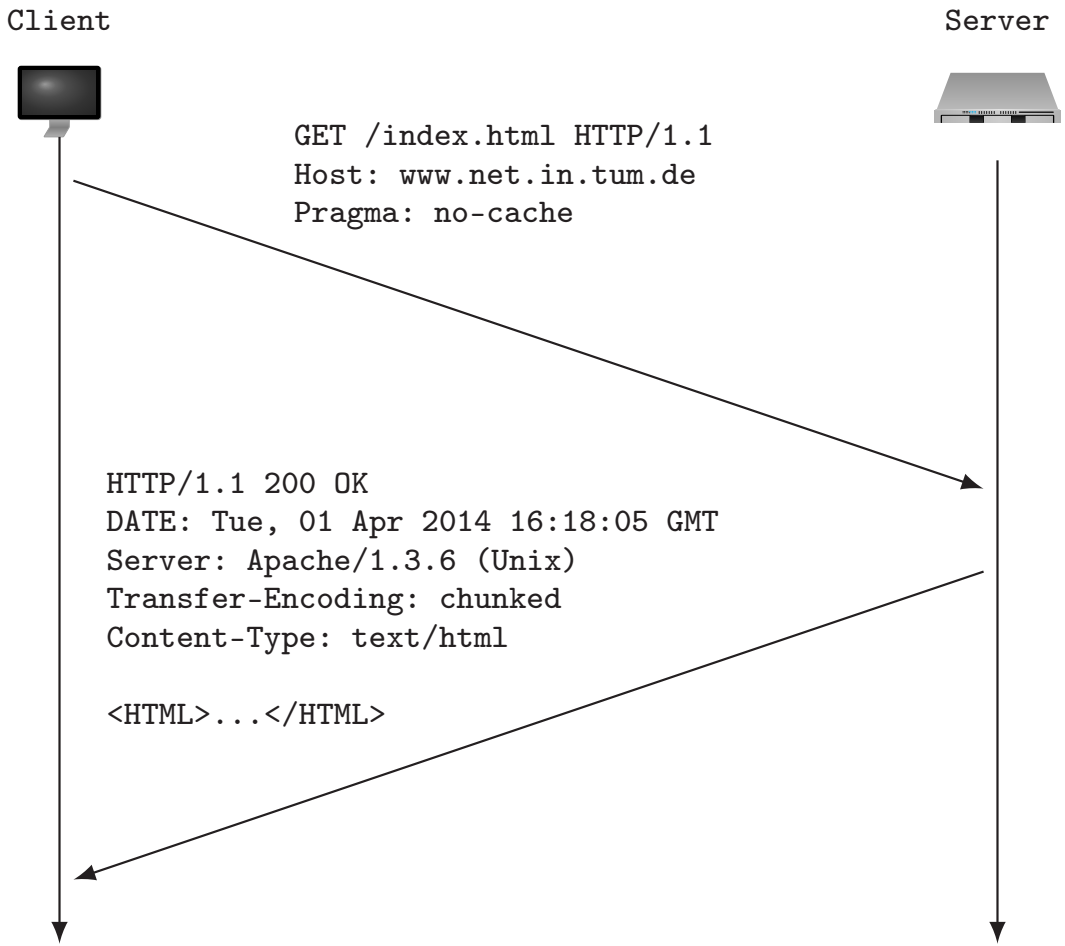
Beispiel


Client
Server
GET /index.html HTTP/1.1
Host: www.net.in.tum.de
Pragma: no-cache

Beispiel


Client
Server
GET /index.html HTTP/1.1
Host: www.net.in.tum.de
Pragma: no-cache
HTTP/1.1 200 OK
DATE: Tue, 01 Apr 2014 16:18:05 GMT
Server: Apache/1.3.6 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html
<HTML>...</HTML>

Beispiel



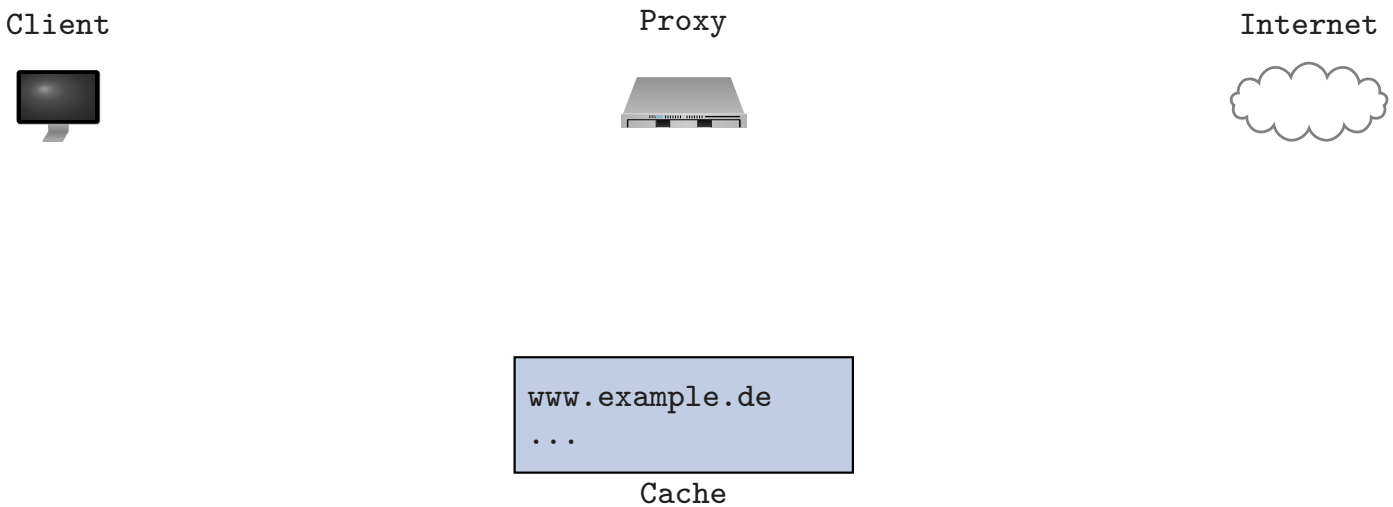
- Die Header-Felder Date und Upgrade (Wechsel des Protokolls) können sowohl im Request als auch im Response verwendet werden.

Gibt es Möglichkeiten Ressourcen schneller zu laden?

- Optimierungen können sowohl auf Seiten des Client, als auch auf Seiten der Server durchgeführt werden.
- Hierfür stehen verschiedene Möglichkeiten zur Verfügung, u. a. Caching mit Hilfe von Proxies, Replikation von Servern (Mirroring) oder sog. Content Delivery Networks.

Definition (Proxy)

Ein Proxy-Server wird zwischen den Client und den Servern postiert und prüft, ob vom Client angefragte Seiten im Cache verfügbar sind oder vom Server geladen werden müssen.

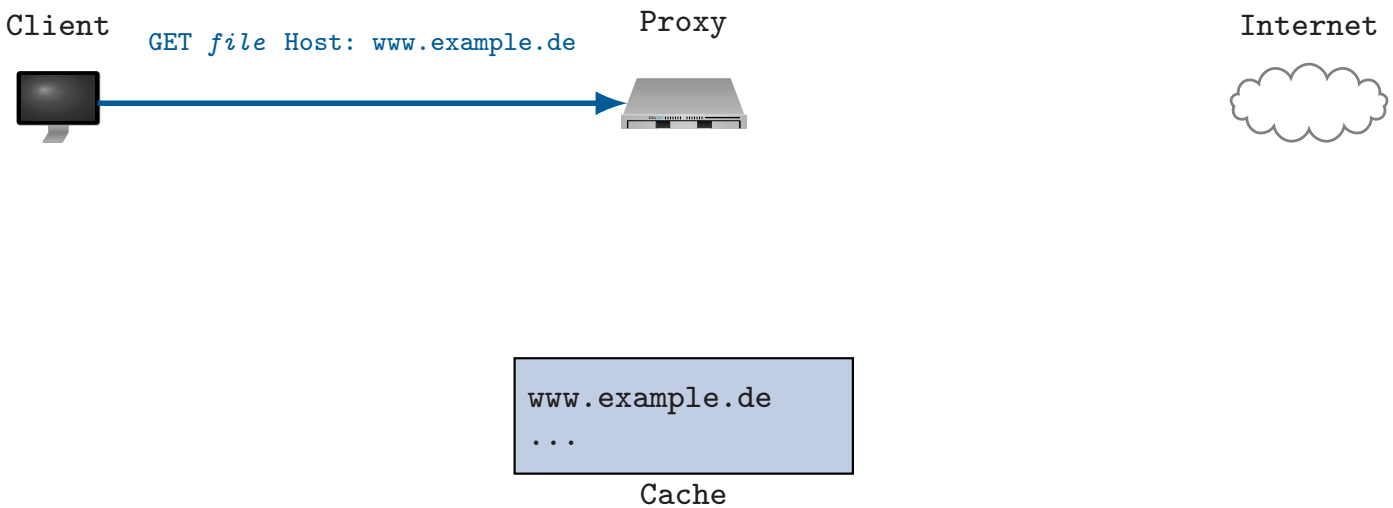


Gibt es Möglichkeiten Ressourcen schneller zu laden?

- Optimierungen können sowohl auf Seiten des Client, als auch auf Seiten der Server durchgeführt werden.
- Hierfür stehen verschiedene Möglichkeiten zur Verfügung, u. a. Caching mit Hilfe von Proxies, Replikation von Servern (Mirroring) oder sog. Content Delivery Networks.

Definition (Proxy)

Ein Proxy-Server wird zwischen den Client und den Servern postiert und prüft, ob vom Client angefragte Seiten im Cache verfügbar sind oder vom Server geladen werden müssen.

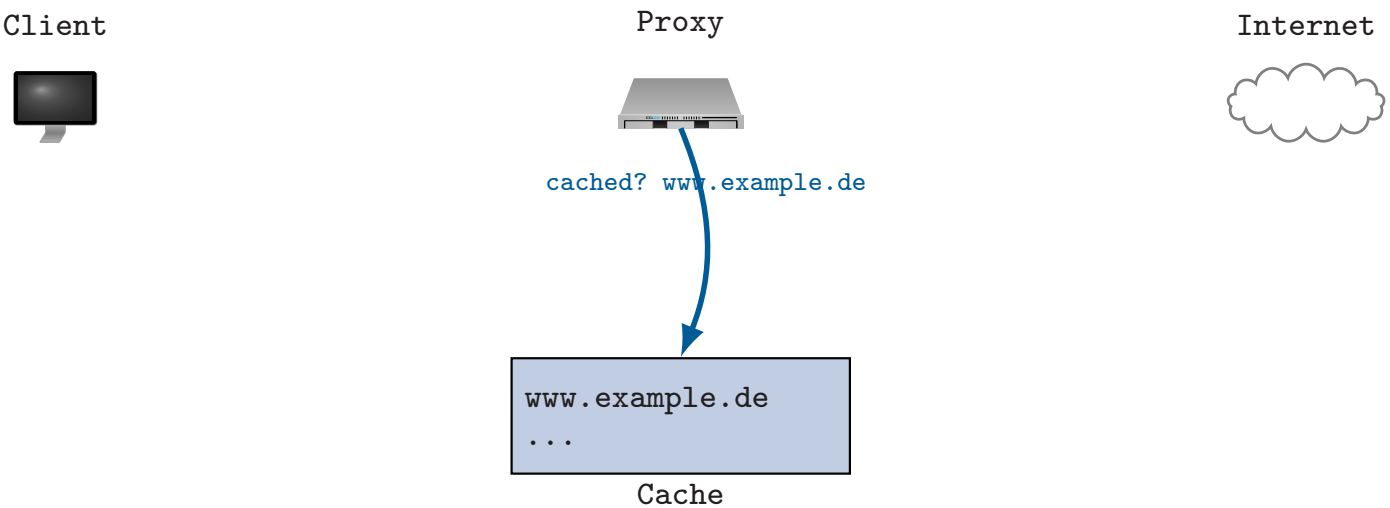


Gibt es Möglichkeiten Ressourcen schneller zu laden?

- Optimierungen können sowohl auf Seiten des Client, als auch auf Seiten der Server durchgeführt werden.
- Hierfür stehen verschiedene Möglichkeiten zur Verfügung, u. a. Caching mit Hilfe von Proxies, Replikation von Servern (Mirroring) oder sog. Content Delivery Networks.

Definition (Proxy)

Ein Proxy-Server wird zwischen den Client und den Servern postiert und prüft, ob vom Client angefragte Seiten im Cache verfügbar sind oder vom Server geladen werden müssen.

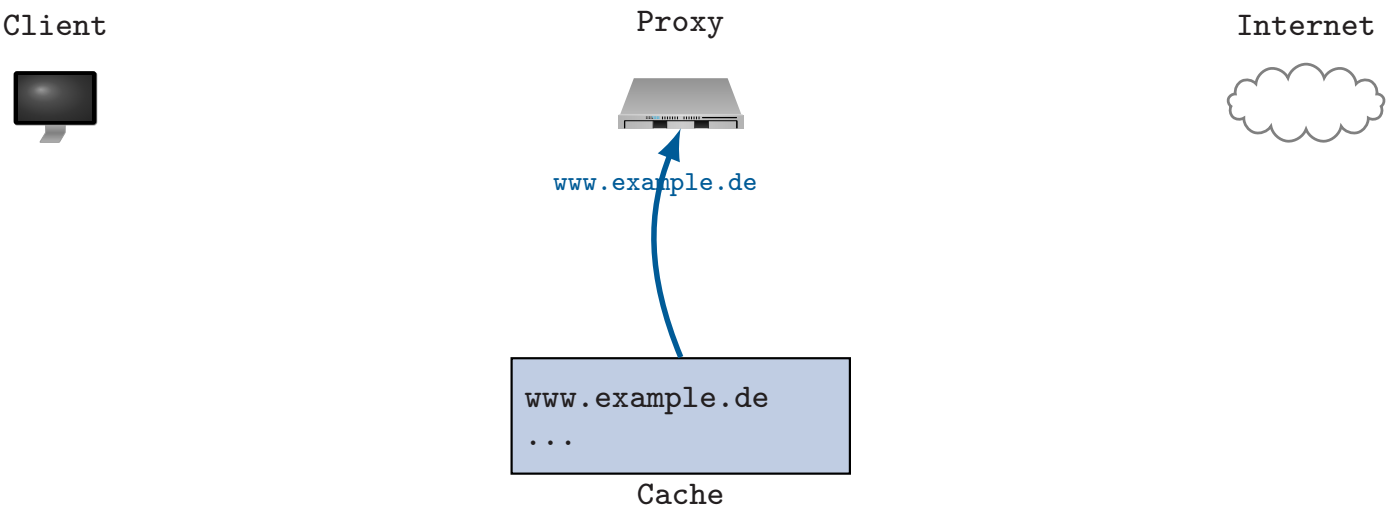


Gibt es Möglichkeiten Ressourcen schneller zu laden?

- Optimierungen können sowohl auf Seiten des Client, als auch auf Seiten der Server durchgeführt werden.
- Hierfür stehen verschiedene Möglichkeiten zur Verfügung, u. a. Caching mit Hilfe von Proxies, Replikation von Servern (Mirroring) oder sog. Content Delivery Networks.

Definition (Proxy)

Ein Proxy-Server wird zwischen den Client und den Servern postiert und prüft, ob vom Client angefragte Seiten im Cache verfügbar sind oder vom Server geladen werden müssen.

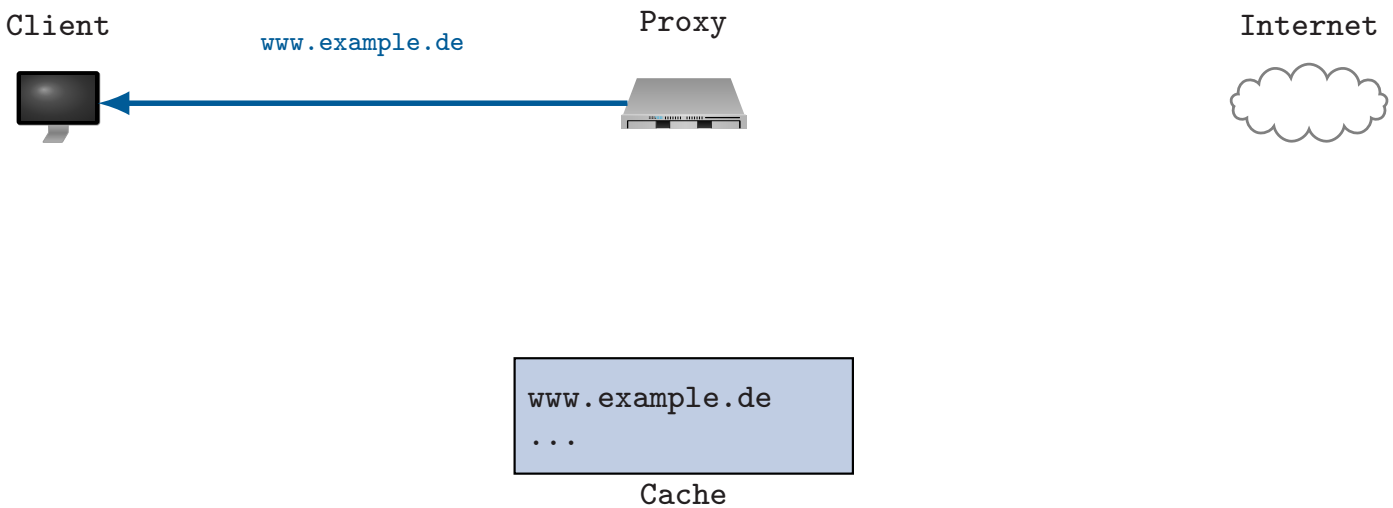


Gibt es Möglichkeiten Ressourcen schneller zu laden?

- Optimierungen können sowohl auf Seiten des Client, als auch auf Seiten der Server durchgeführt werden.
- Hierfür stehen verschiedene Möglichkeiten zur Verfügung, u. a. Caching mit Hilfe von Proxies, Replikation von Servern (Mirroring) oder sog. Content Delivery Networks.

Definition (Proxy)

Ein Proxy-Server wird zwischen den Client und den Servern postiert und prüft, ob vom Client angefragte Seiten im Cache verfügbar sind oder vom Server geladen werden müssen.

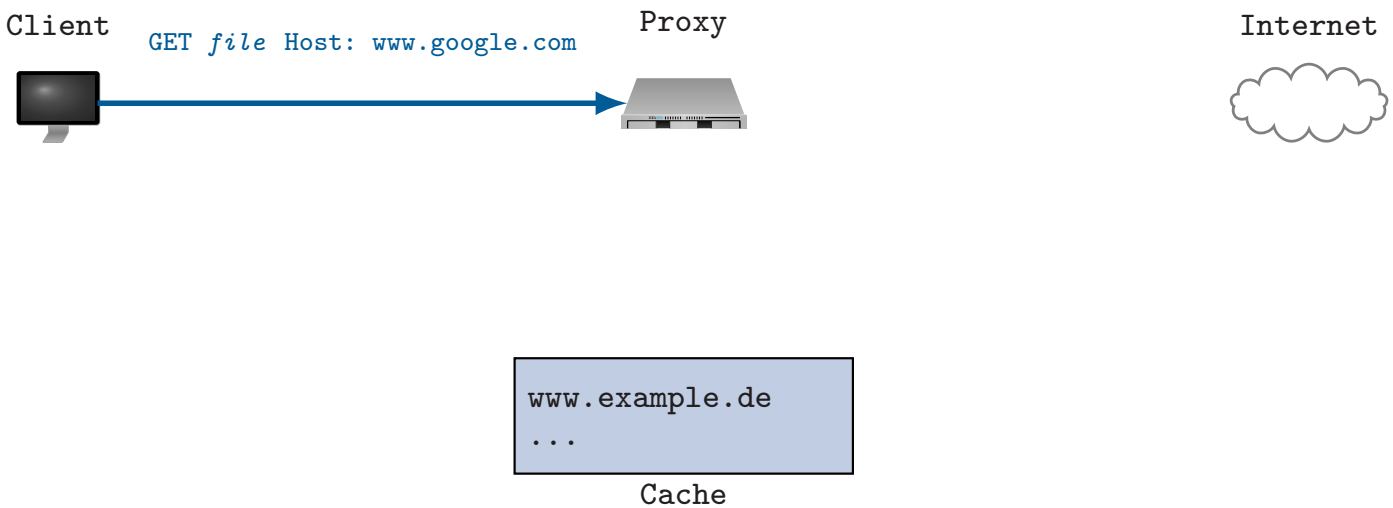


Gibt es Möglichkeiten Ressourcen schneller zu laden?

- Optimierungen können sowohl auf Seiten des Client, als auch auf Seiten der Server durchgeführt werden.
- Hierfür stehen verschiedene Möglichkeiten zur Verfügung, u. a. Caching mit Hilfe von Proxies, Replikation von Servern (Mirroring) oder sog. Content Delivery Networks.

Definition (Proxy)

Ein Proxy-Server wird zwischen den Client und den Servern postiert und prüft, ob vom Client angefragte Seiten im Cache verfügbar sind oder vom Server geladen werden müssen.

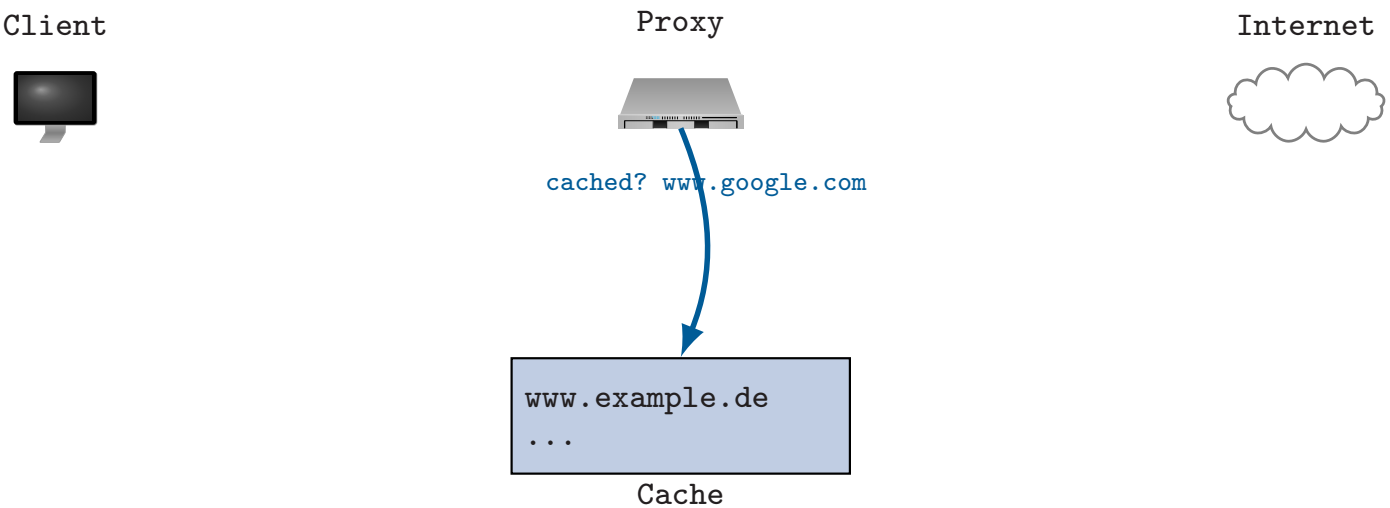


Gibt es Möglichkeiten Ressourcen schneller zu laden?

- Optimierungen können sowohl auf Seiten des Client, als auch auf Seiten der Server durchgeführt werden.
- Hierfür stehen verschiedene Möglichkeiten zur Verfügung, u. a. Caching mit Hilfe von Proxies, Replikation von Servern (Mirroring) oder sog. Content Delivery Networks.

Definition (Proxy)

Ein Proxy-Server wird zwischen den Client und den Servern postiert und prüft, ob vom Client angefragte Seiten im Cache verfügbar sind oder vom Server geladen werden müssen.

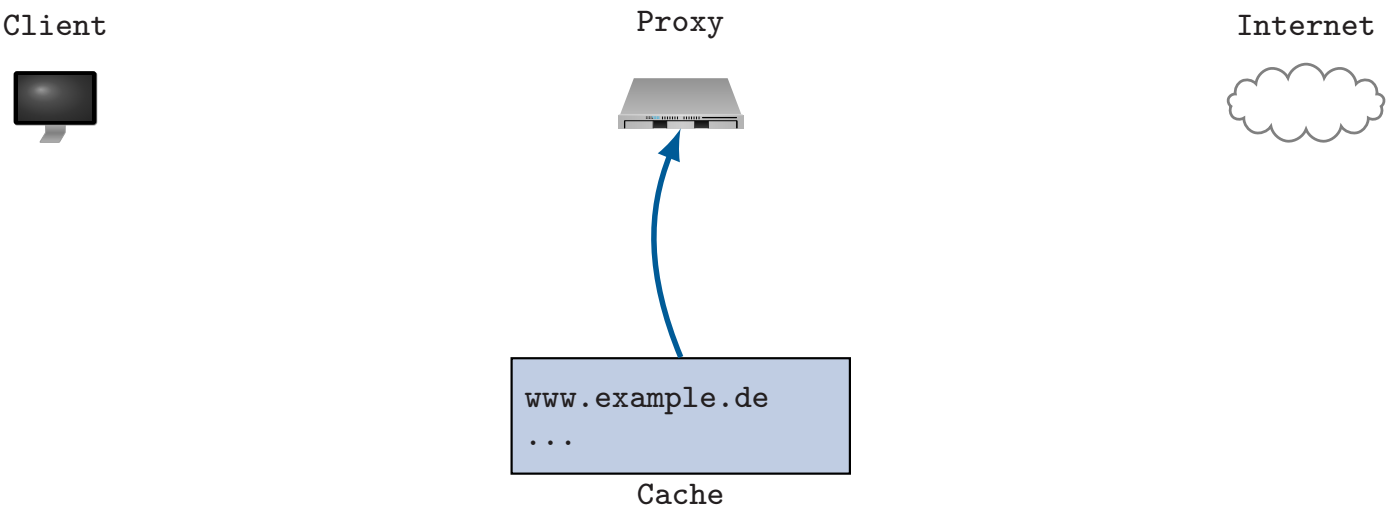


Gibt es Möglichkeiten Ressourcen schneller zu laden?

- Optimierungen können sowohl auf Seiten des Client, als auch auf Seiten der Server durchgeführt werden.
- Hierfür stehen verschiedene Möglichkeiten zur Verfügung, u. a. Caching mit Hilfe von Proxies, Replikation von Servern (Mirroring) oder sog. Content Delivery Networks.

Definition (Proxy)

Ein Proxy-Server wird zwischen den Client und den Servern postiert und prüft, ob vom Client angefragte Seiten im Cache verfügbar sind oder vom Server geladen werden müssen.

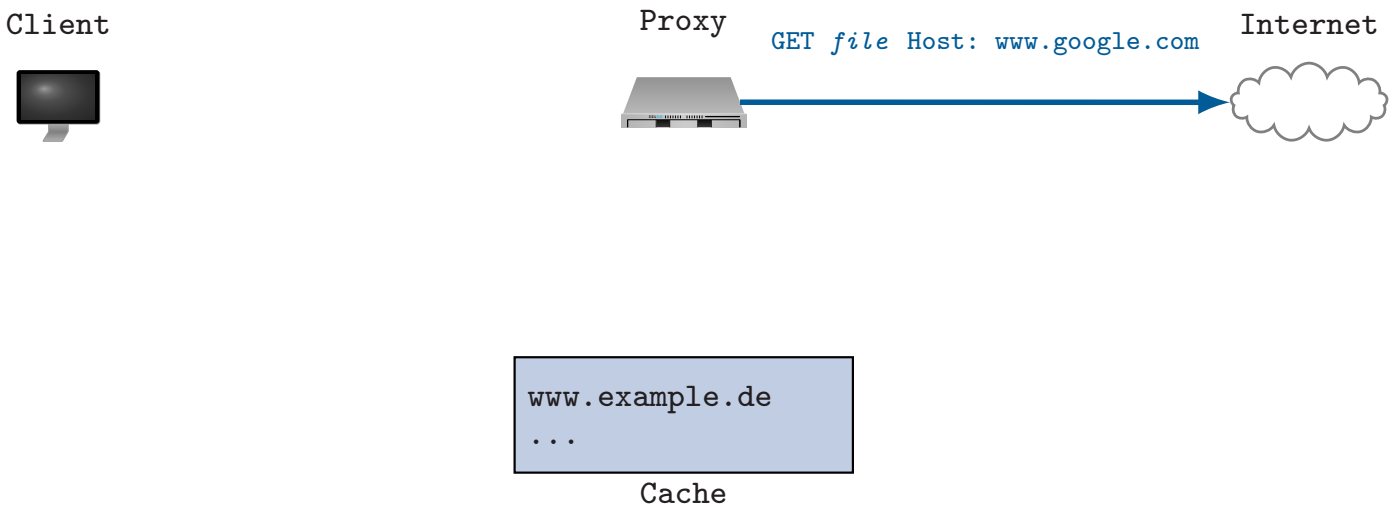


Gibt es Möglichkeiten Ressourcen schneller zu laden?

- Optimierungen können sowohl auf Seiten des Client, als auch auf Seiten der Server durchgeführt werden.
- Hierfür stehen verschiedene Möglichkeiten zur Verfügung, u. a. Caching mit Hilfe von Proxies, Replikation von Servern (Mirroring) oder sog. Content Delivery Networks.

Definition (Proxy)

Ein Proxy-Server wird zwischen den Client und den Servern postiert und prüft, ob vom Client angefragte Seiten im Cache verfügbar sind oder vom Server geladen werden müssen.

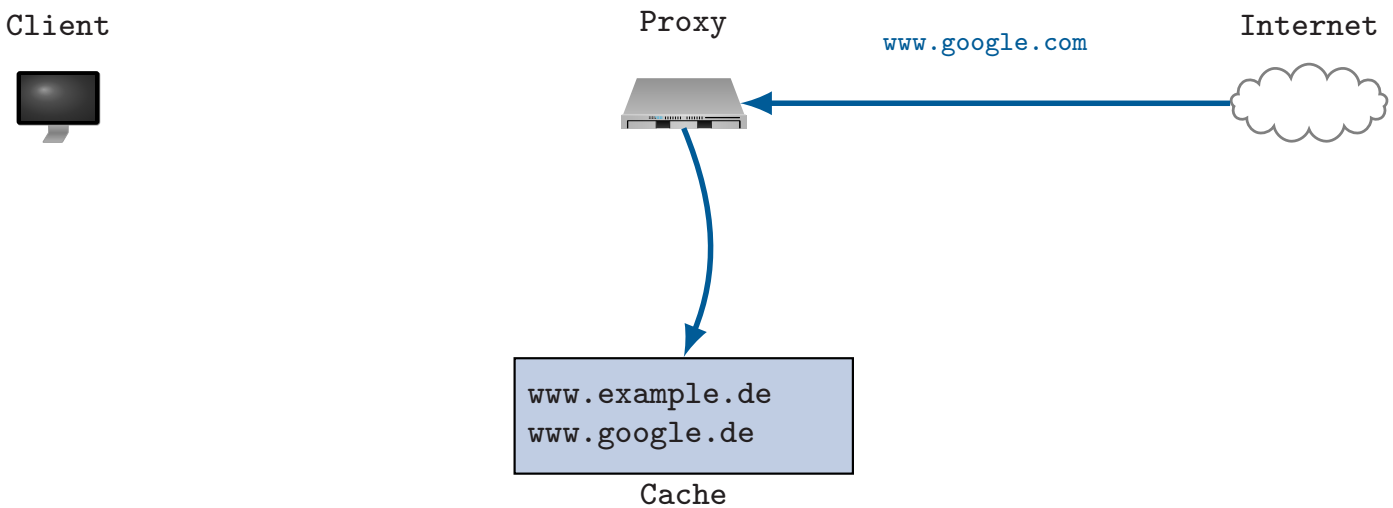


Gibt es Möglichkeiten Ressourcen schneller zu laden?

- Optimierungen können sowohl auf Seiten des Client, als auch auf Seiten der Server durchgeführt werden.
- Hierfür stehen verschiedene Möglichkeiten zur Verfügung, u. a. Caching mit Hilfe von Proxies, Replikation von Servern (Mirroring) oder sog. Content Delivery Networks.

Definition (Proxy)

Ein Proxy-Server wird zwischen den Client und den Servern postiert und prüft, ob vom Client angefragte Seiten im Cache verfügbar sind oder vom Server geladen werden müssen.

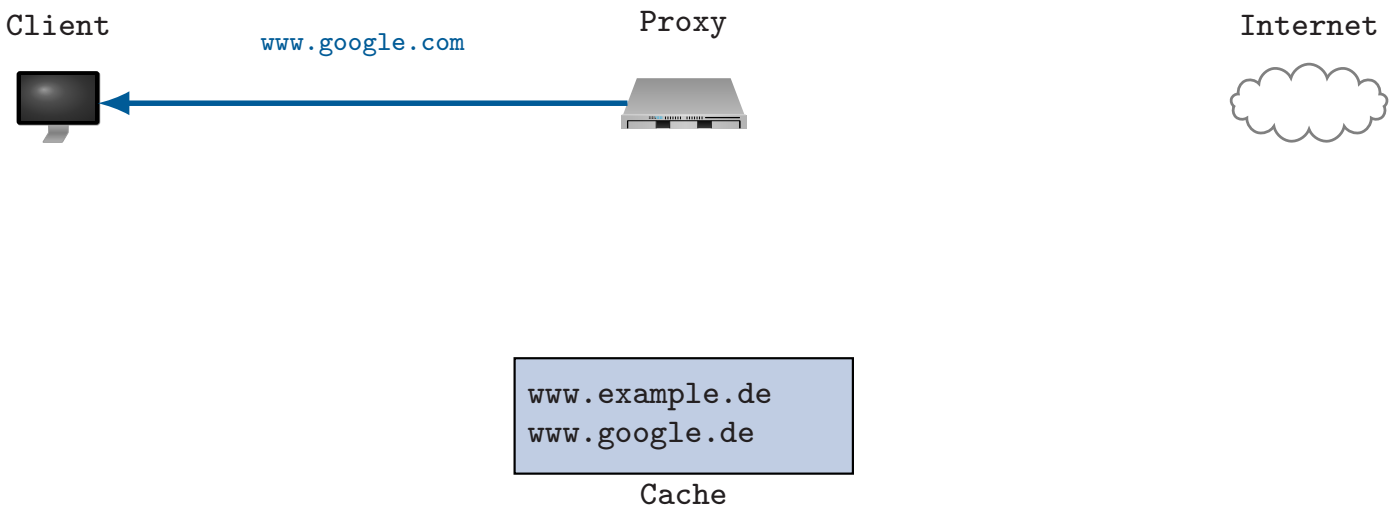


Gibt es Möglichkeiten Ressourcen schneller zu laden?

- Optimierungen können sowohl auf Seiten des Client, als auch auf Seiten der Server durchgeführt werden.
- Hierfür stehen verschiedene Möglichkeiten zur Verfügung, u. a. Caching mit Hilfe von Proxies, Replikation von Servern (Mirroring) oder sog. Content Delivery Networks.

Definition (Proxy)

Ein Proxy-Server wird zwischen den Client und den Servern postiert und prüft, ob vom Client angefragte Seiten im Cache verfügbar sind oder vom Server geladen werden müssen.



Electronic-Mail (e-Mail)

- Die ersten Vorschläge zur Standardisierung wurden bereits 1982 in den RFCs 821 (Übertragungsprotokoll) und 822 (Nachrichtenformat) vorgelegt.
- Endgültig zum Standard wurden die entsprechenden Revisionen 2821 und 2822 erklärt.

Definition (e-Mail-Architektur)

Es sind zwei Teilsysteme von Bedeutung: die User-Agents werden von den Benutzern verwendet, um e-Mails zu lesen und zu verwalten. Die Message-Transfer-Agents leiten Nachrichten der Sender bis zum Ziel weiter.

Die folgenden **5 Grundfunktionen** sollten von allen e-Mail-Systemen unterstützt werden:

- Erstellung (Composition)
- Übertragung (Transfer)
- Berichterstellung (Reporting)
- Anzeige (Display)
- Verwertung (Disposition)

e-Mail - Format

e-Mail - Format

```
MAIL FROM: <foo@net.in.tum.de>
RCPT TO: <ba@net.in.tum.de>
```

Envelope für MTA

- Der Envelope enthält alle Informationen für den Transport der e-Mail zum Empfänger.

e-Mail - Format

```
MAIL FROM: <foo@net.in.tum.de>
RCPT TO: <ba@net.in.tum.de>
```

Envelope für MTA

```
To: <ba@online.de>
CC: <andere@online.de>
Bcc: <unsichtbar@online.de>
From: <foo@net.in.tum.de>
Sender: <ghostwriter@net.in.tum.de>
Received: from smtp1.informatik.tu-muenchen.de ([131.159.0.8]) by
(...) for <ba@online.de>; Tue, 01 Apr 2014 10:15:04 +0200
Return-Path: <foo@net.in.tum.de>
Date: Tue, 01 Apr 2014 10:15:03 +0200
Message-ID: <01ab91cf4d82$7b206430$71627390$@net.in.tum.de>
Reply-to: <foo@net.in.tum.de>
In-Reply-To: <99ab91cf4d82$7b206430$71627390$@net.in.tum.de>
References: <99ab91cf4d82$7b206430$71627390$@net.in.tum.de>
<88ab91cf4d82$7b206430$71627390$@net.in.tum.de>
Subject: Re: Re: Status Update
```

Header für UA

- Der Envelope enthält alle Informationen für den Transport der e-Mail zum Empfänger.
- Der Header ist optional und enthält Informationen über den Absender bzw. Empfänger.

e-Mail - Format

```
MAIL FROM: <foo@net.in.tum.de>
RCPT TO: <ba@net.in.tum.de>
```

Envelope für MTA

```
To: <ba@online.de>
CC: <andere@online.de>
Bcc: <unsichtbar@online.de>
From: <foo@net.in.tum.de>
Sender: <ghostwriter@net.in.tum.de>
Received: from smtp1.informatik.tu-muenchen.de ([131.159.0.8]) by
(...) for <ba@online.de>; Tue, 01 Apr 2014 10:15:04 +0200
Return-Path: <foo@net.in.tum.de>
Date: Tue, 01 Apr 2014 10:15:03 +0200
Message-ID: <01ab91cf4d82$7b206430$71627390$@net.in.tum.de>
Reply-to: <foo@net.in.tum.de>
In-Reply-To: <99ab91cf4d82$7b206430$71627390$@net.in.tum.de>
References: <99ab91cf4d82$7b206430$71627390$@net.in.tum.de>
<88ab91cf4d82$7b206430$71627390$@net.in.tum.de>
Subject: Re: Re: Status Update
```

Header für UA

```
Terminbestätigung
Montag bis Mittwoch ...
```

Body

- Der Envelope enthält alle Informationen für den Transport der e-Mail zum Empfänger.
- Der Header ist optional und enthält Informationen über den Absender bzw. Empfänger.
- Der Body beinhaltet den eigentlichen Inhalt.

Multipurpose Internet Mail Extensions (MIME)

- **Problem:** e-Mails wurden als reine Textnachrichten im ASCII-Format entworfen.
- **Lösung:** Erweiterung der Strukturen für den Nachrichtentext und Definition der Kodierungsregeln für nicht-ASCII-Nachrichten mit Hilfe 5 zusätzlicher Header.

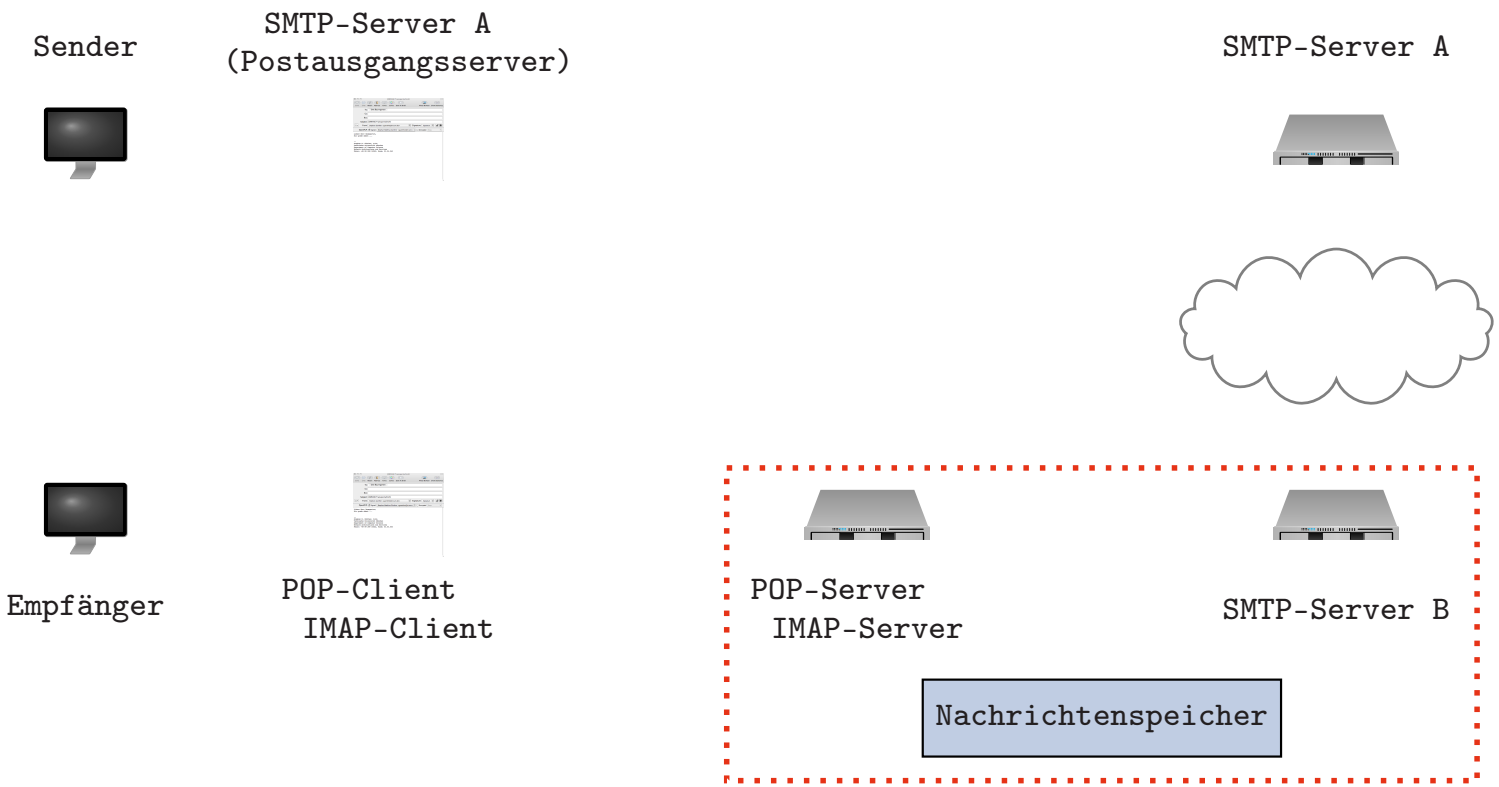
Header	Beschreibung
MIME-Version	verwendete MIME-Version
Content-Description	Beschreibung des Inhalts in ASCII-Zeichen
Content-ID	Eindeutiger Bezeichner im gleichen Format wie die Message-ID
Content-Transfer-Encoding	für die Übertragung verwendete Kodierung (5 Schemata, z. B. base64)
Content-Type	Art des Nachrichteninhalts

Der Content-Type besteht aus der Typangabe sowie dem Untertyp, welche durch einen "/" getrennt werden, z. B. image/gif. Einige Typen sind in der folgenden Tabelle gelistet.

Typ	Untertypen	Beschreibung
Text	plain, enriched	unformatierter bzw. einfach formatierter ASCII-Text
Image	gif, jpeg	Standbild im GIF- oder JPEG-Format
Audio	basic	Audiodateien
Video	mpeg	Video im MPEG-Format

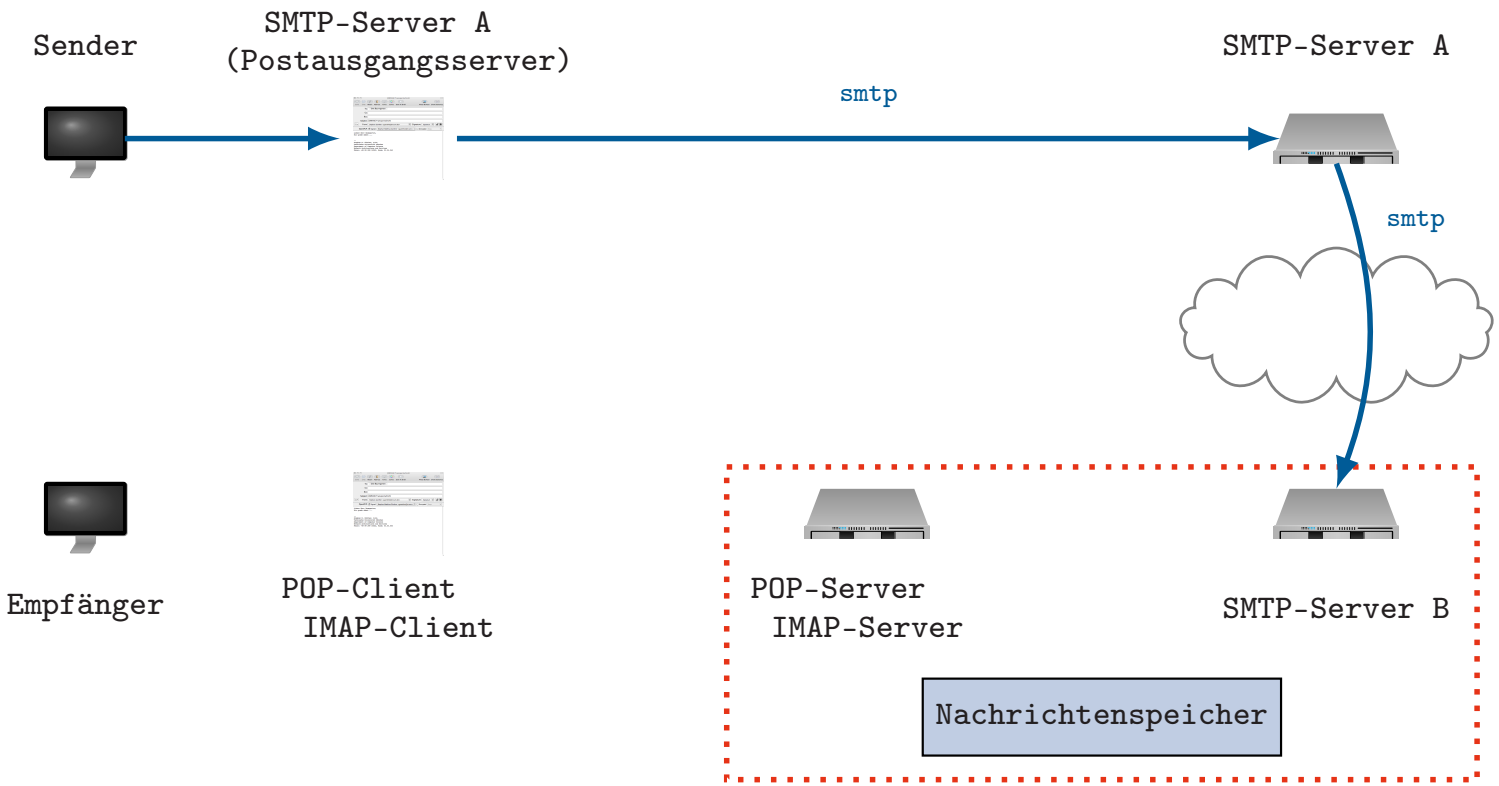
Mailserver-Architektur

- SMTP wird als Protokoll zum Versand von e-Mail zwischen MTAs und von einem UA zum entsprechenden MTA verwendet.
- Nachrichten können dann mittels POP3 oder IMAP vom entprechenden SMTP-Server abgeholt bzw. dort verwaltet werden.



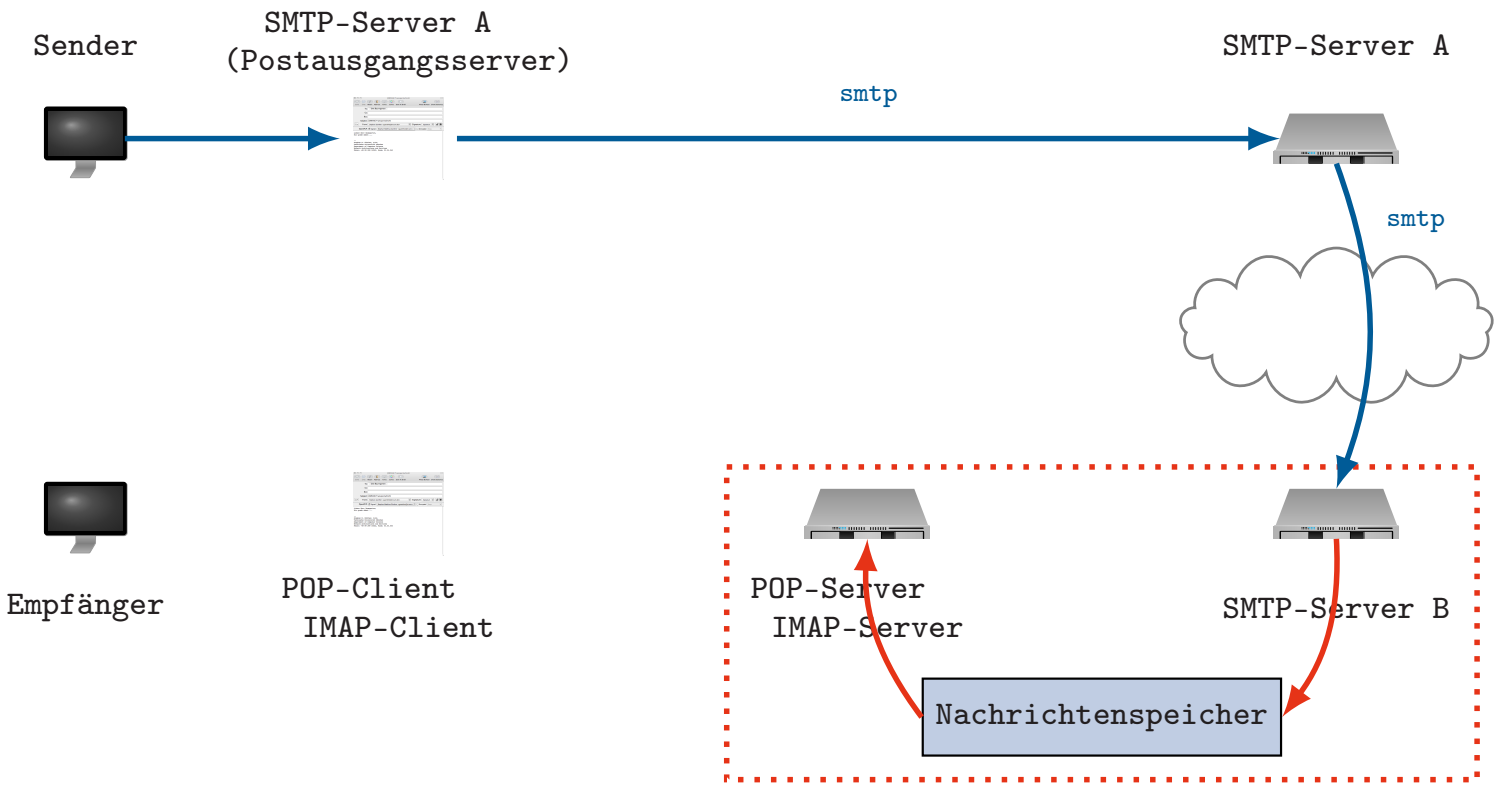
Mailserver-Architektur

- SMTP wird als Protokoll zum Versand von e-Mail zwischen MTAs und von einem UA zum entsprechenden MTA verwendet.
- Nachrichten können dann mittels POP3 oder IMAP vom entprechenden SMTP-Server abgeholt bzw. dort verwaltet werden.



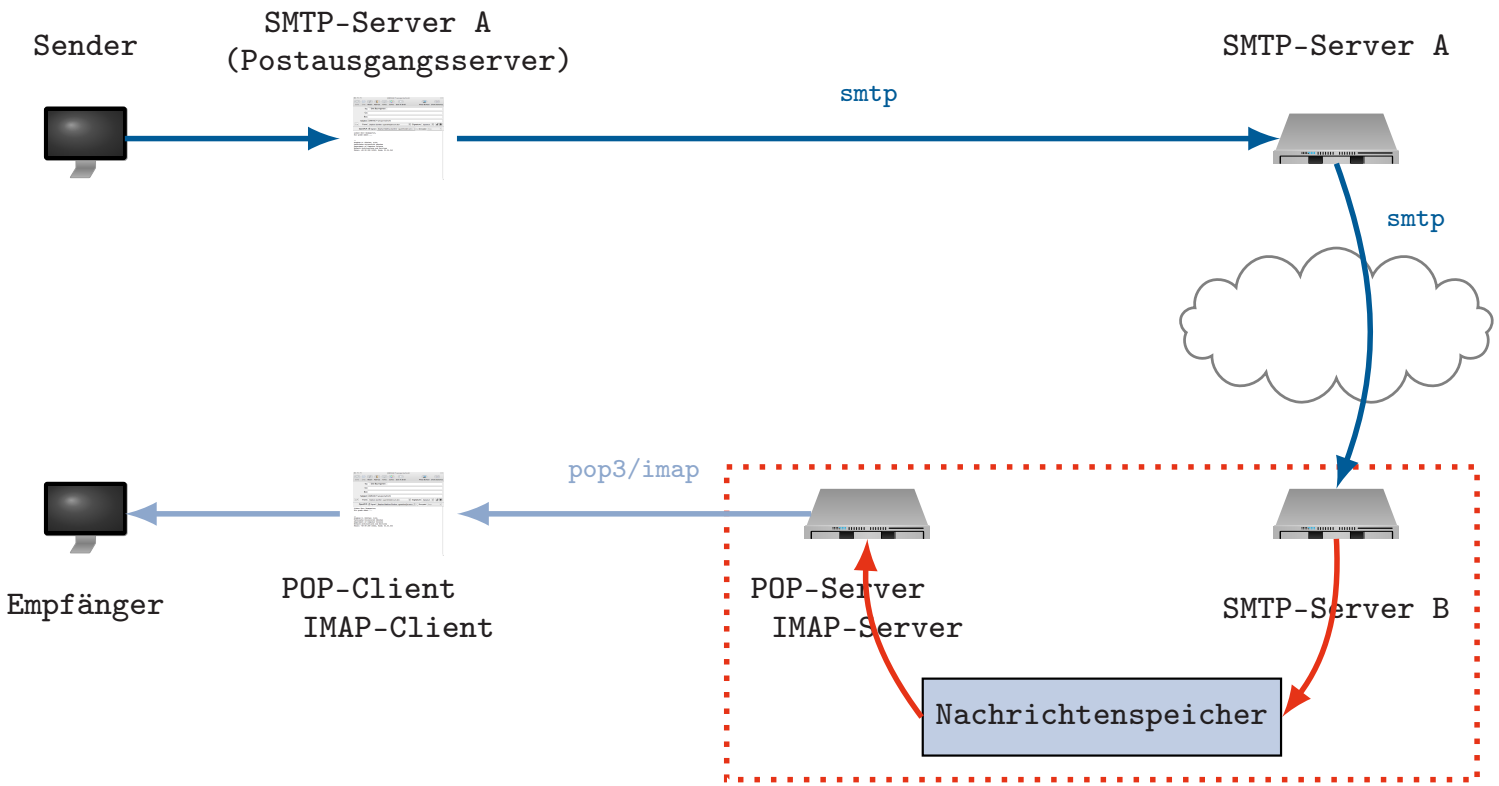
Mailserver-Architektur

- SMTP wird als Protokoll zum Versand von e-Mail zwischen MTAs und von einem UA zum entsprechenden MTA verwendet.
- Nachrichten können dann mittels POP3 oder IMAP vom entprechenden SMTP-Server abgeholt bzw. dort verwaltet werden.



Mailserver-Architektur

- SMTP wird als Protokoll zum Versand von e-Mail zwischen MTAs und von einem UA zum entsprechenden MTA verwendet.
- Nachrichten können dann mittels POP3 oder IMAP vom entprechenden SMTP-Server abgeholt bzw. dort verwaltet werden.



Simple Mail Transfer Protocol (SMTP)

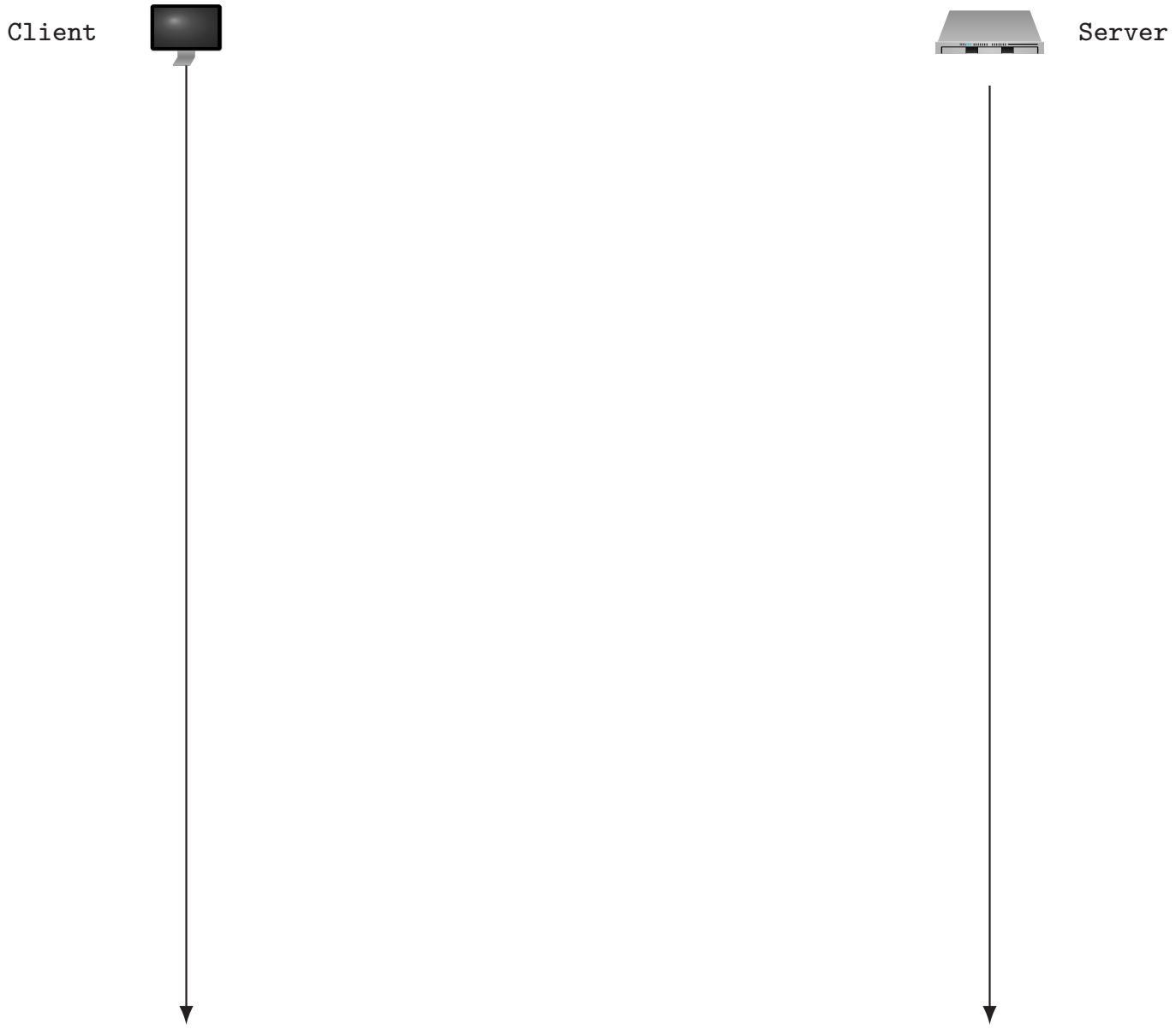
- SMTP ist ein zeichenorientiertes Protokoll und benutzt standardmäßig den Port 25.
- Port 587 wird verwendet, wenn eine Authentifizerung der Benutzer durchgeführt wird. Der Austausch zwischen MTAs findet weiterhin über Port 25 statt.
- Die Kommunikation zwischen SMTP-Entitäten erfolgt über einen einfachen Kommandosatz.

Kommando	Beschreibung
HELO	Start der SMTP Sitzung und Identifikation der Teilnehmer
MAIL	Start der Mailübertragung
RCPT	Angabe der Empfängeradresse, Kommando kann mehrfach ausgeführt werden
DATA	Übermittlung der eigentlichen Nachricht, beendet mittels "."
RSET	Abbruch der Übertragung, Verbindung bleibt bestehen
VRTY oder EXPN	Prüfen der Empfängeradresse
NOOP	Antwort vom Server erzwingen, z. B. um einen Timeout zu vermeiden
QUIT	Beenden der Verbindung

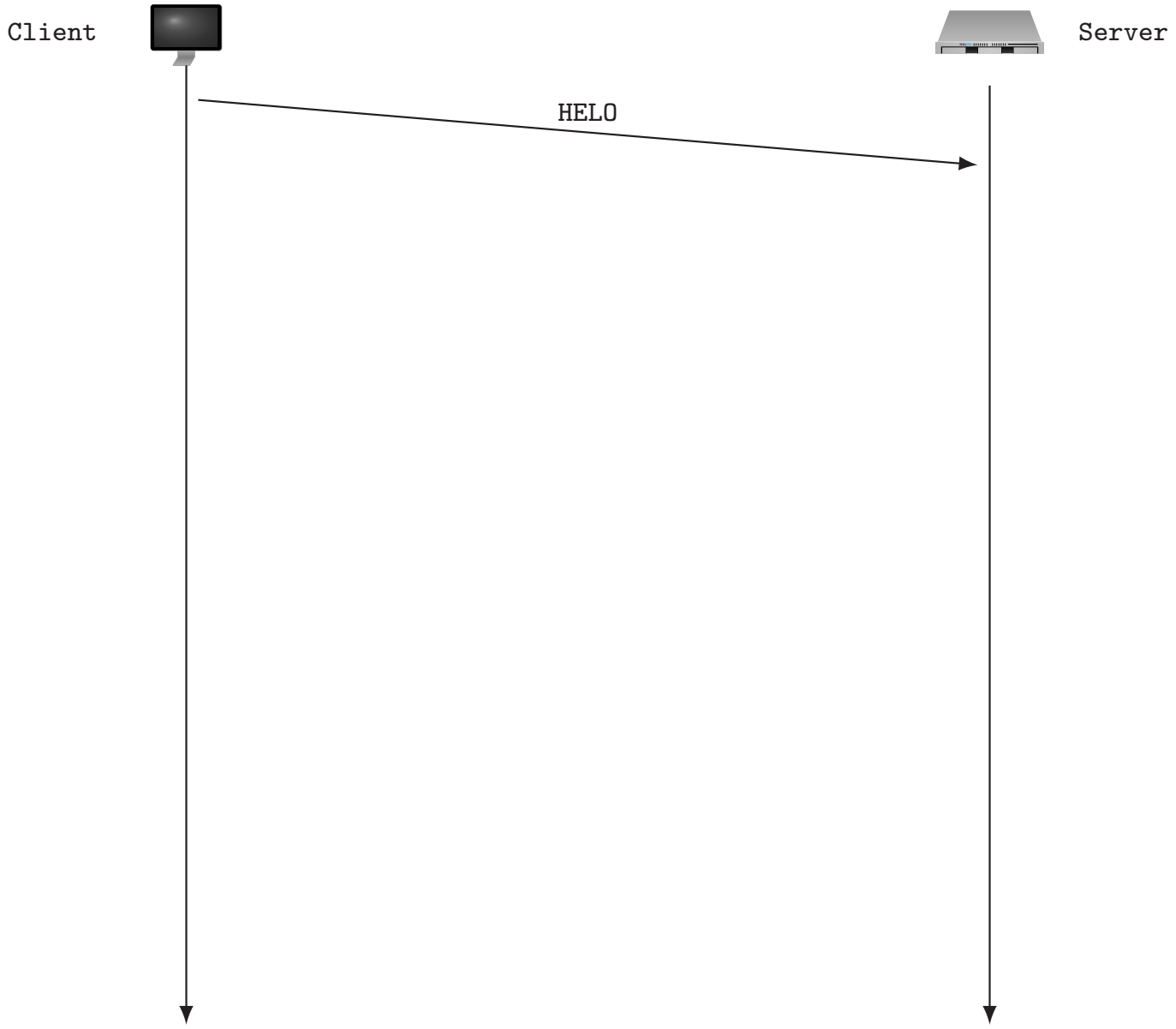
Eine Antwort des Servers besteht aus einem 3-stelligen Statuscode, sowie der Klartextmeldung.

Code	Bedeutung	Code	Bedeutung
250	*OK*, Kommando wurde erfolgreich ausgeführt	354	Starte Mail Input
421	Service nicht verfügbar	451	Ausführungsfehler und Abbruch
503	Falsche Kommandoreihenfolge	554	Transaktion fehlgeschalgen

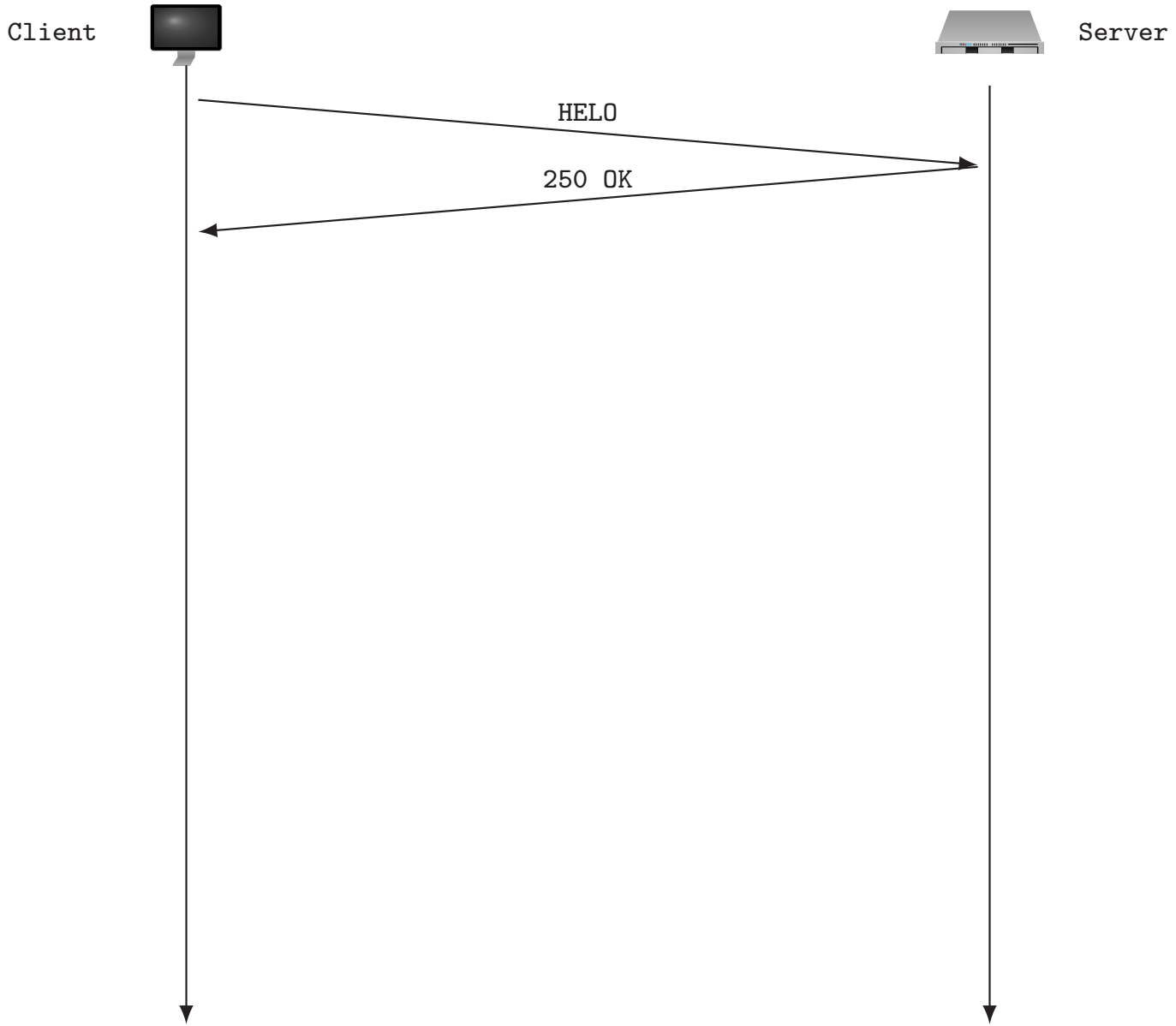
Ablaufbeispiel



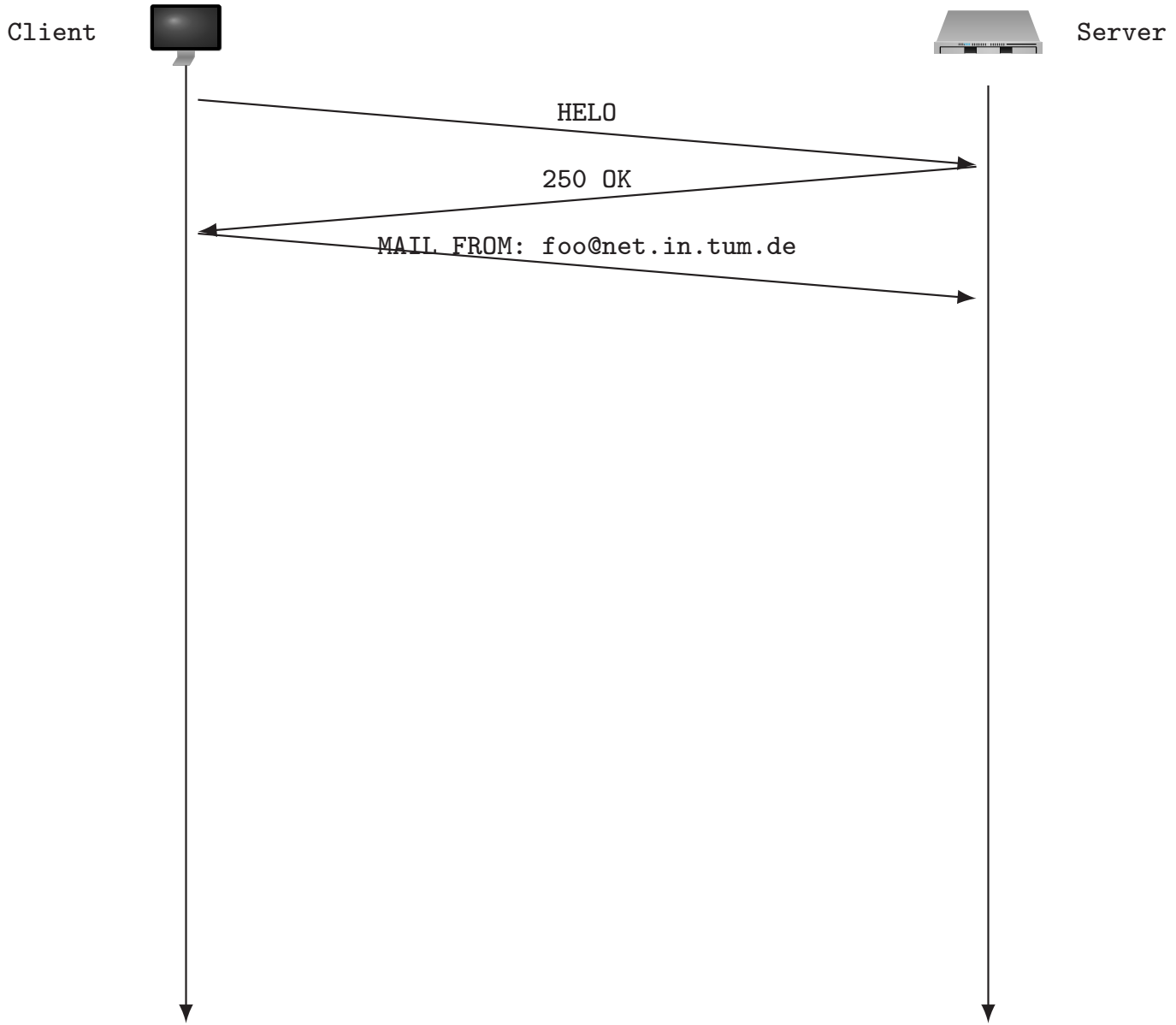
Ablaufbeispiel



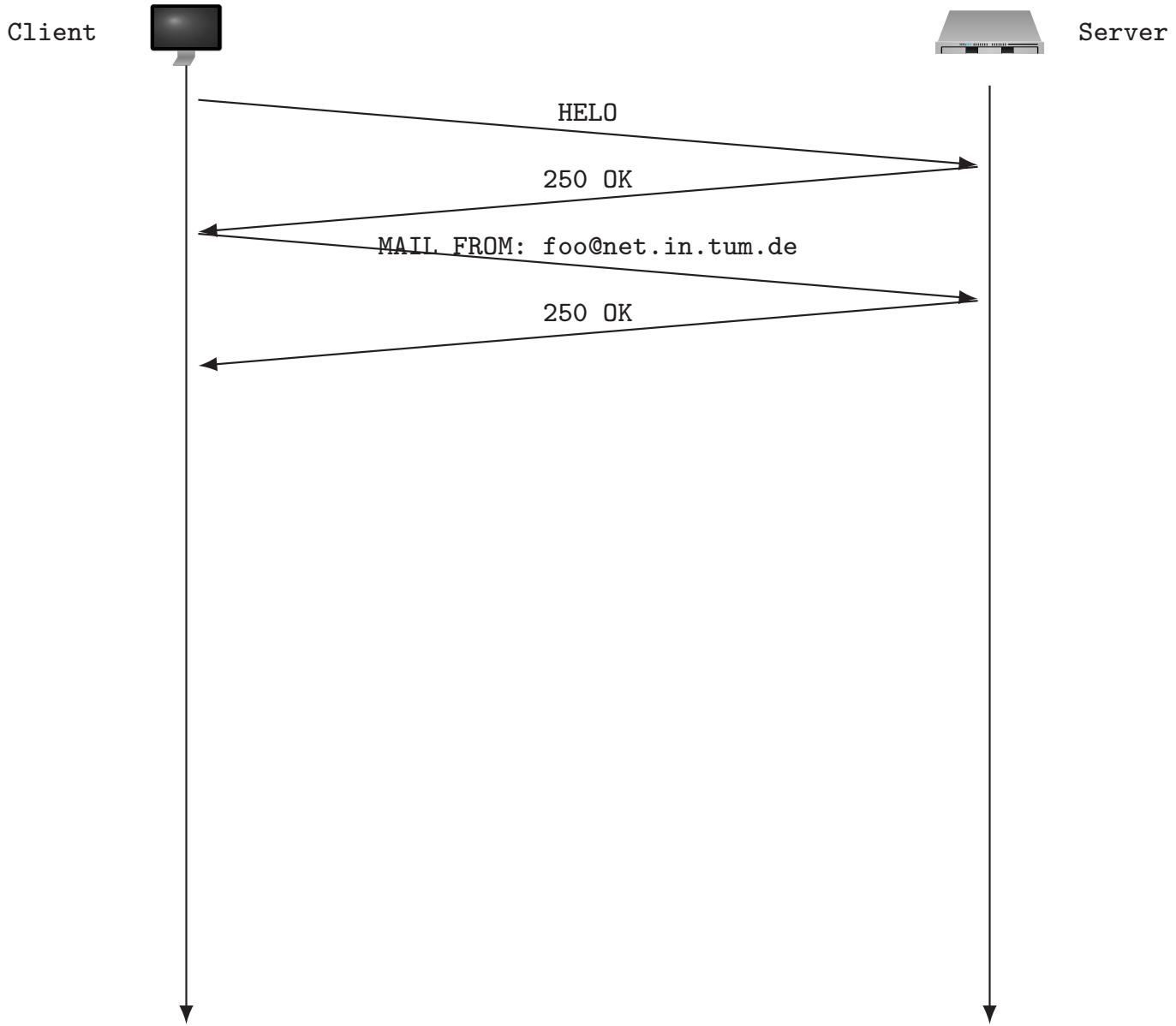
Ablaufbeispiel



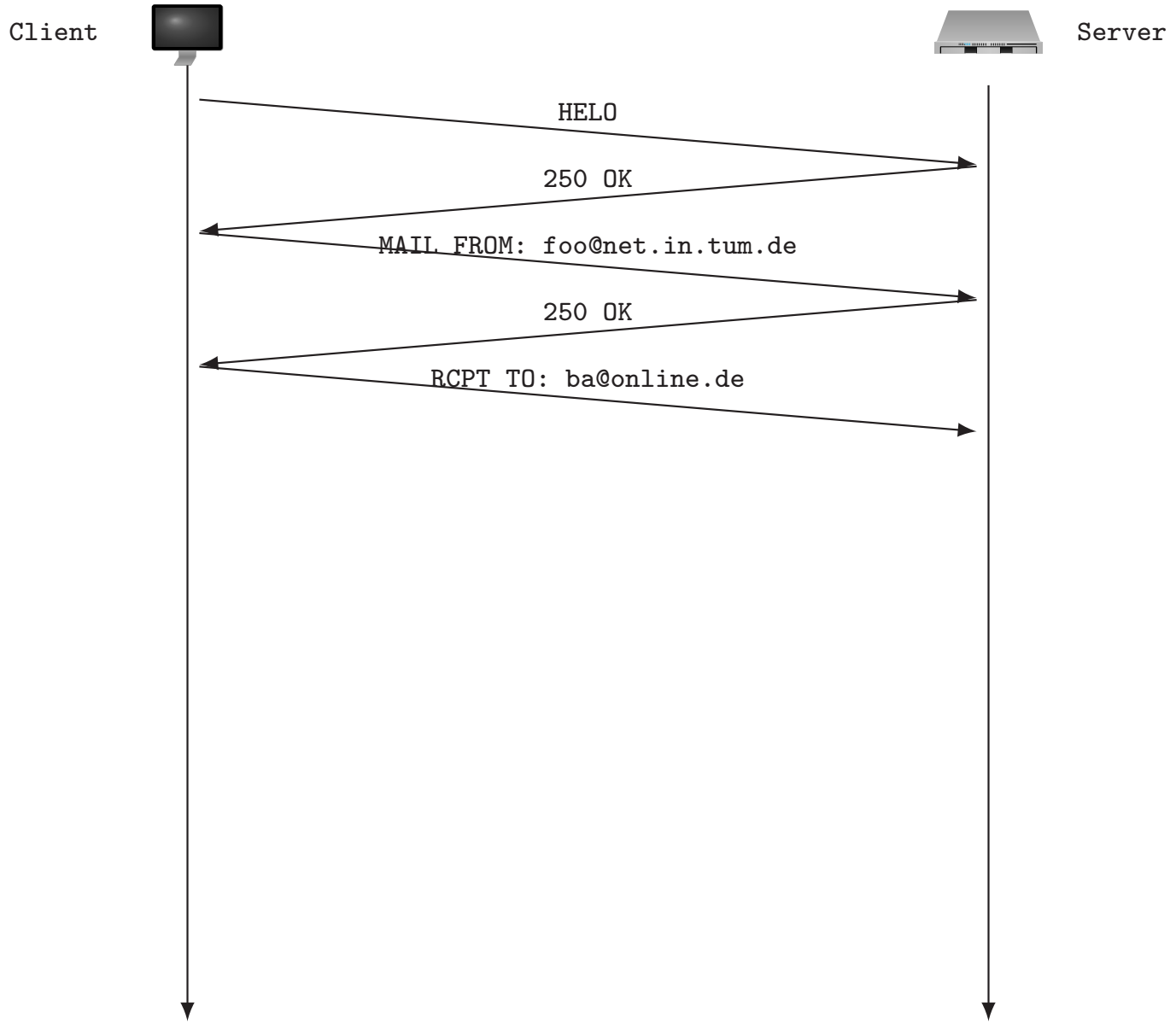
Ablaufbeispiel



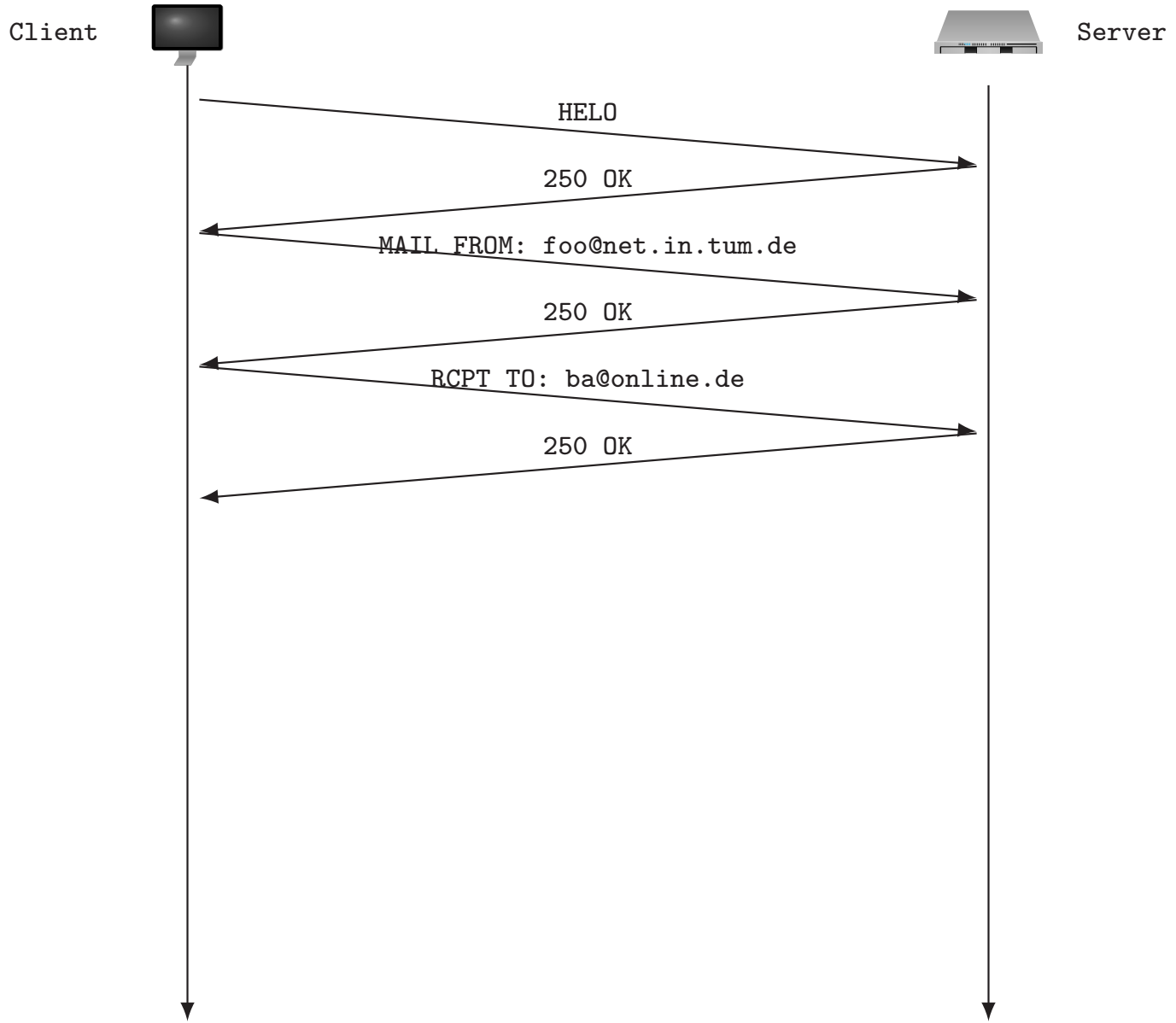
Ablaufbeispiel



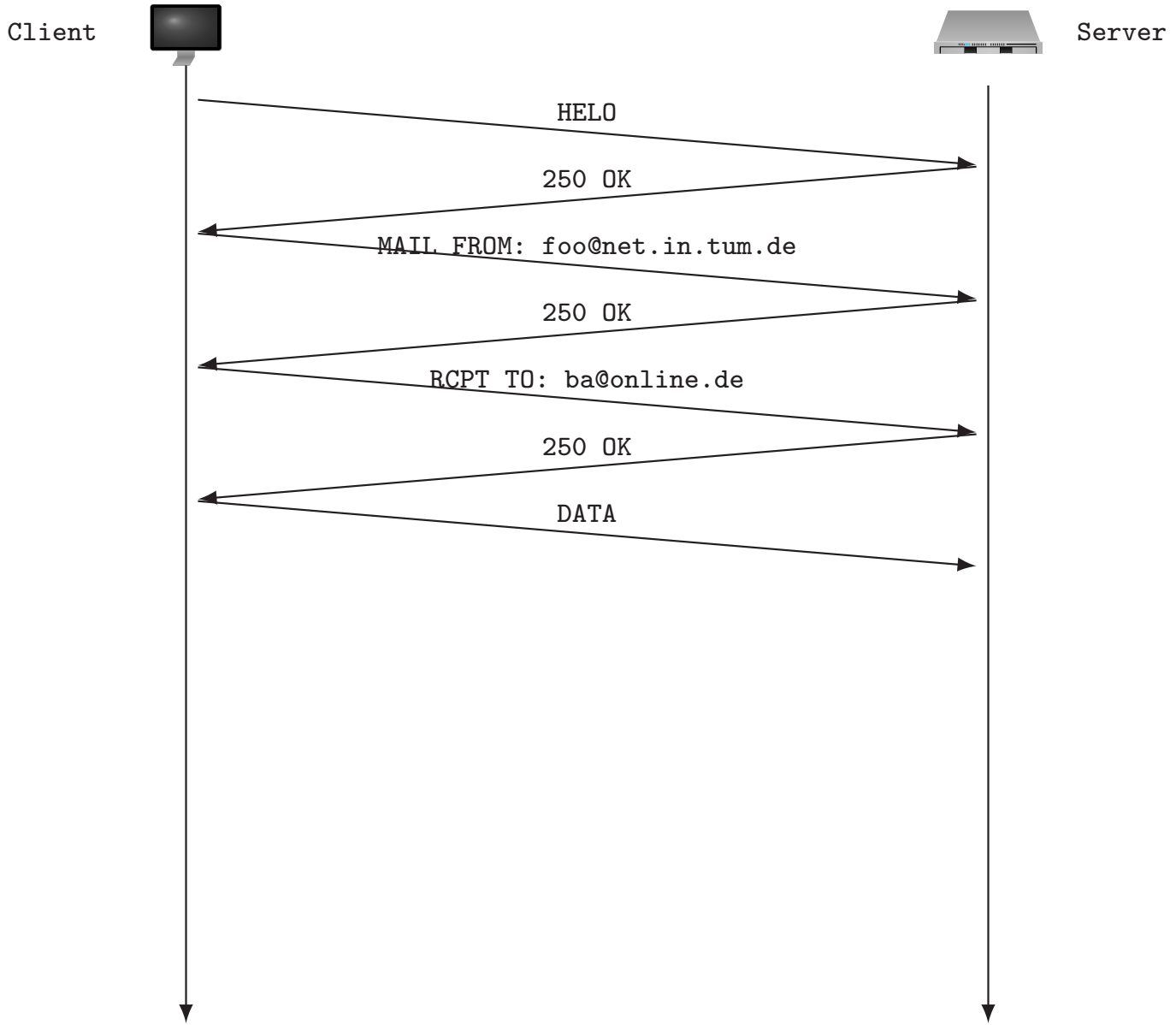
Ablaufbeispiel



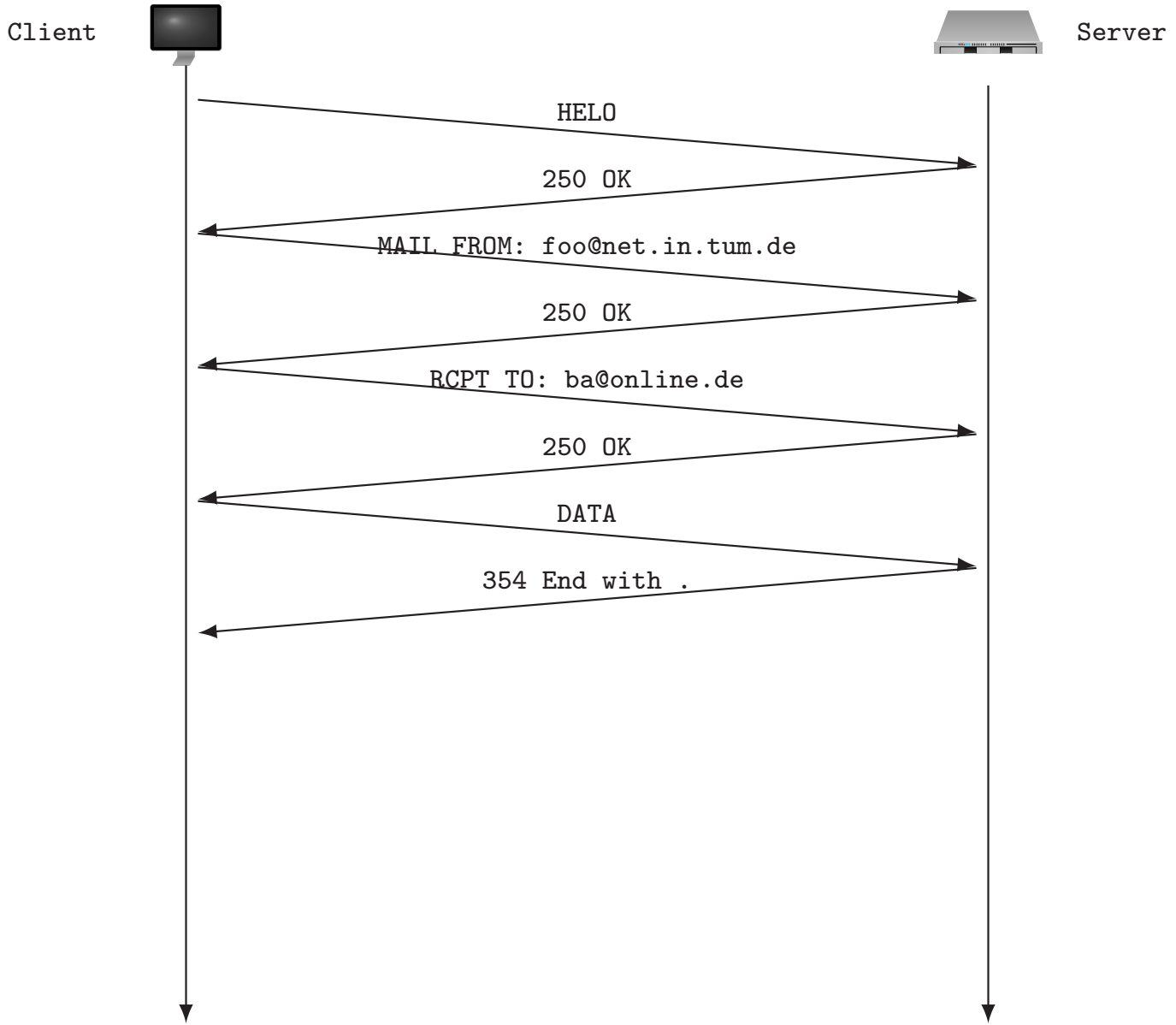
Ablaufbeispiel



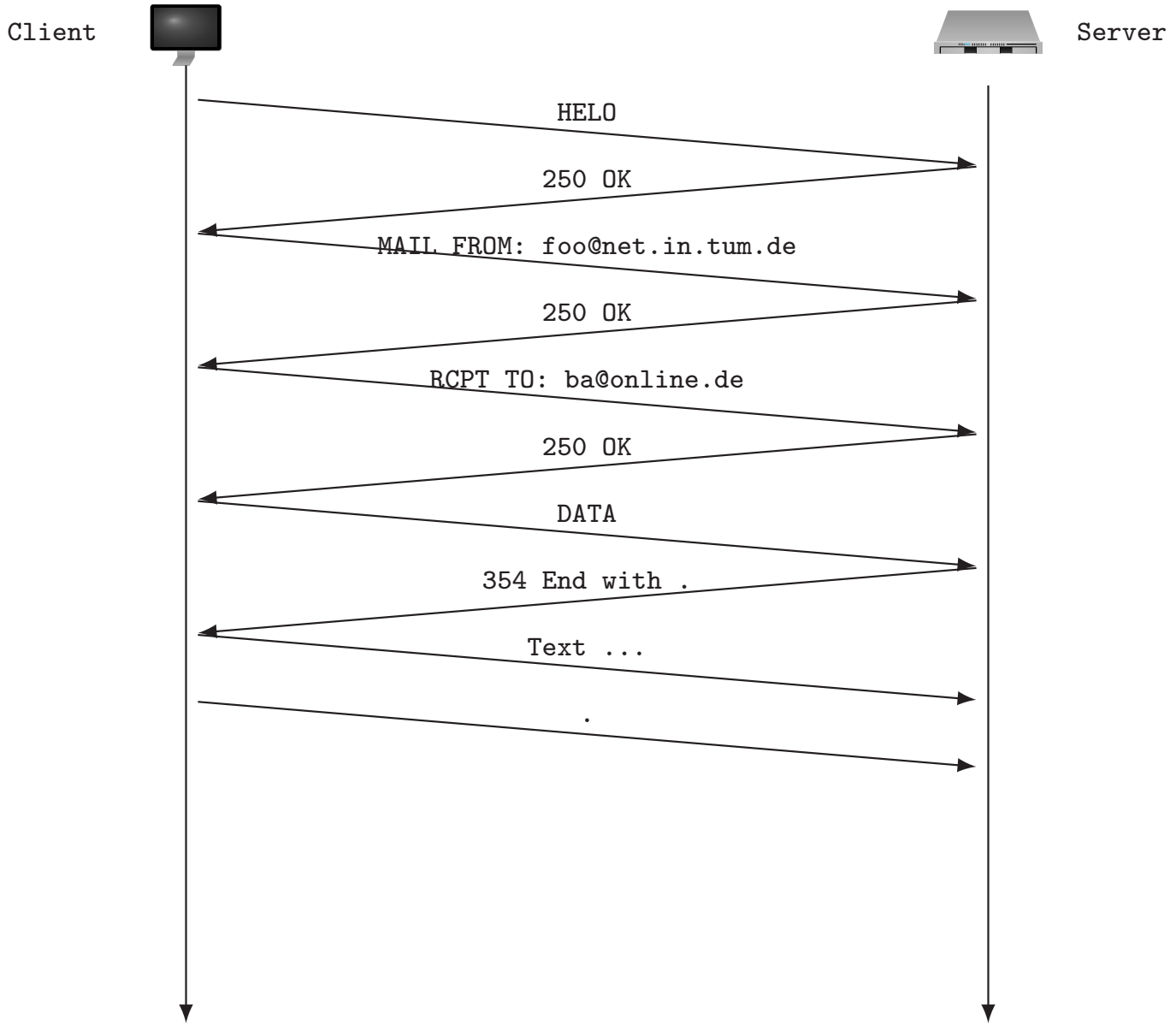
Ablaufbeispiel



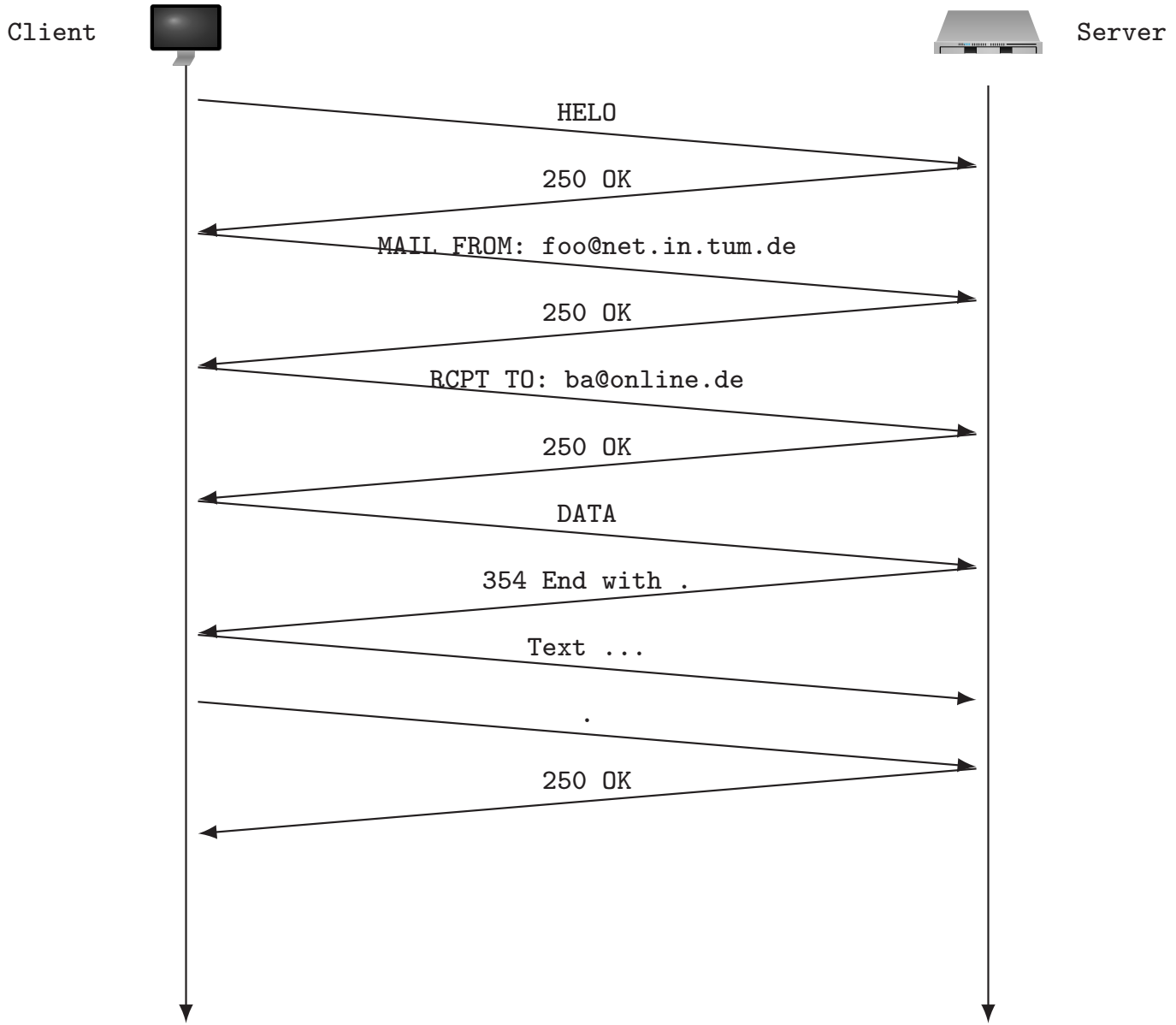
Ablaufbeispiel



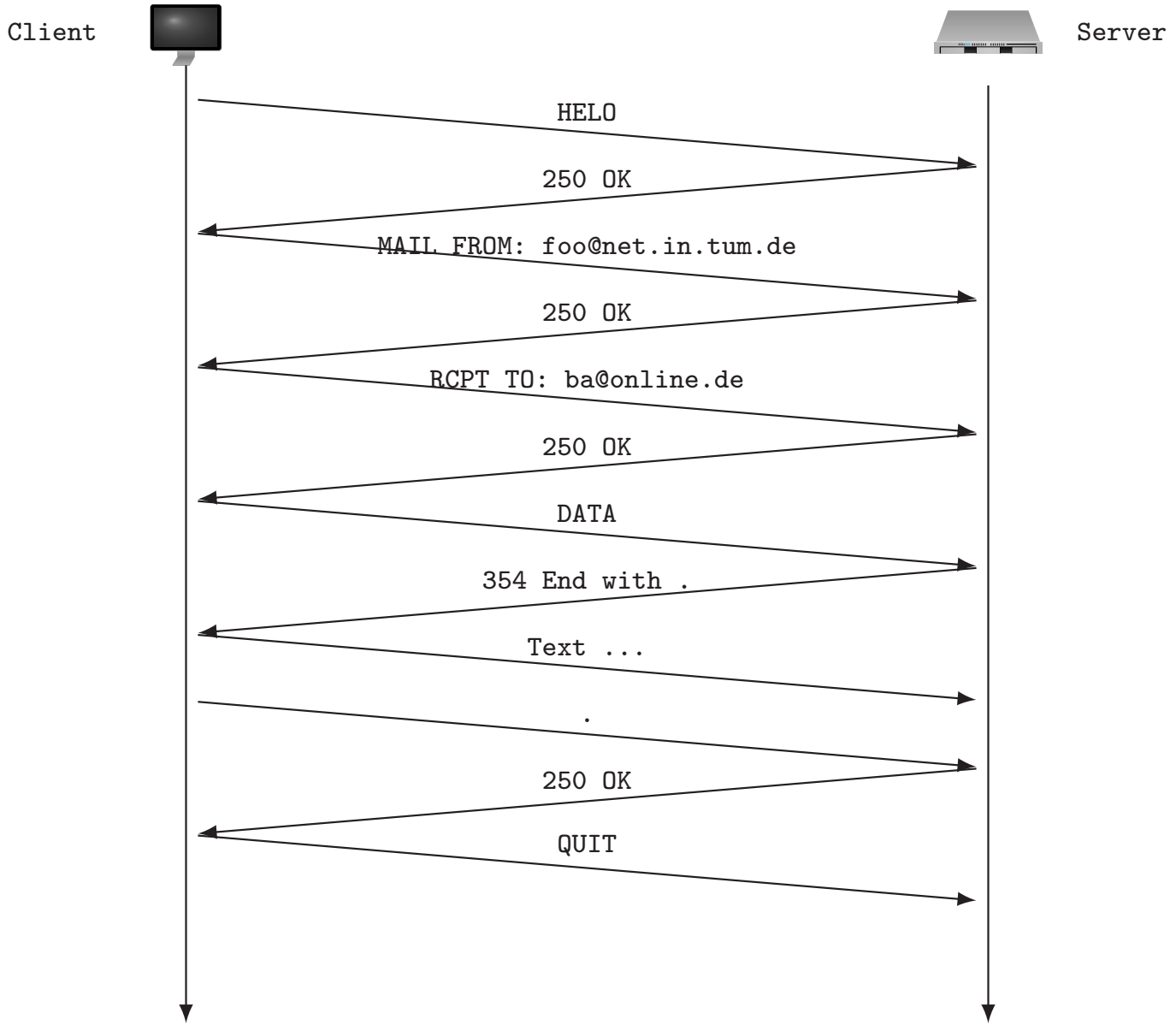
Ablaufbeispiel



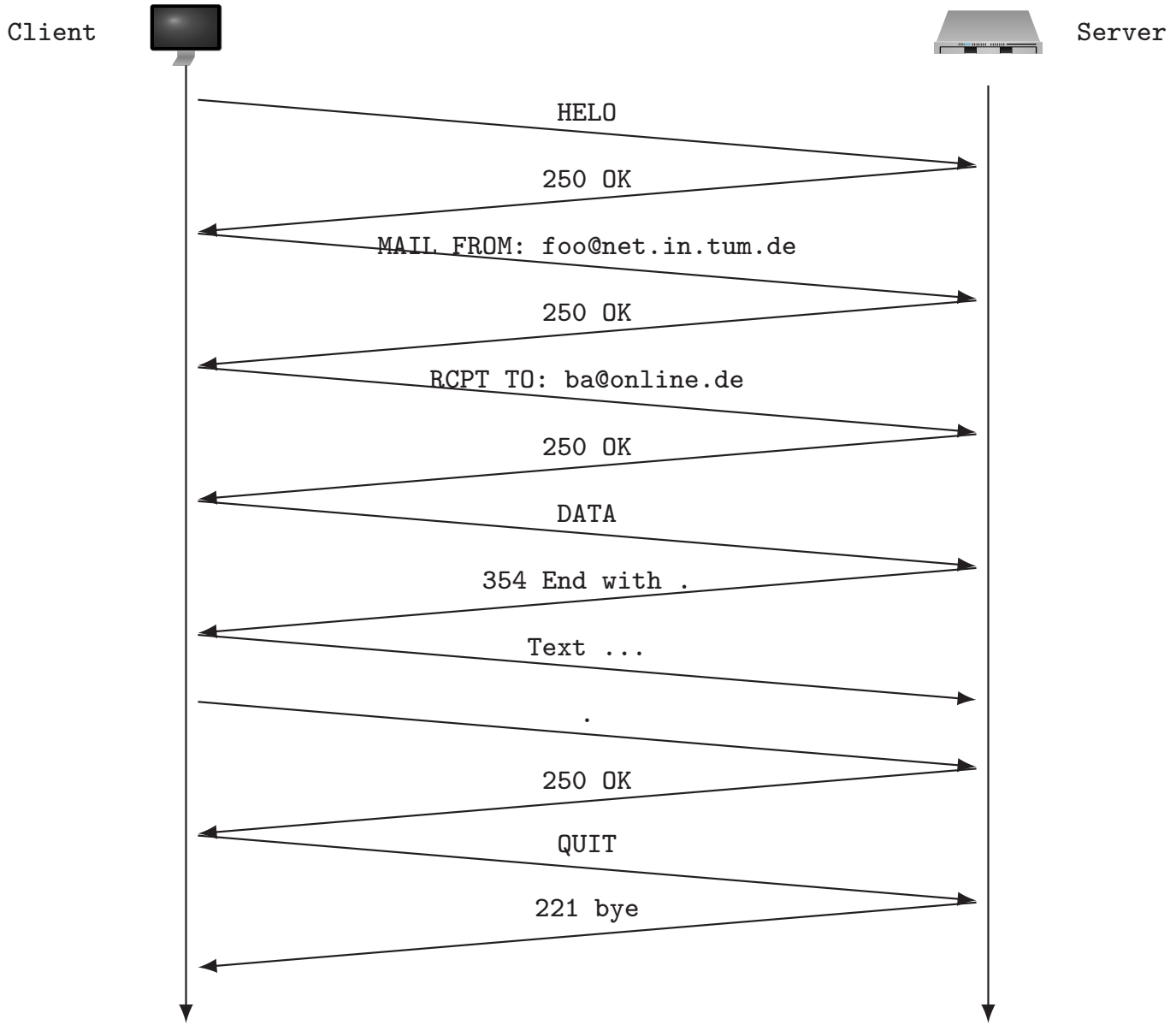
Ablaufbeispiel



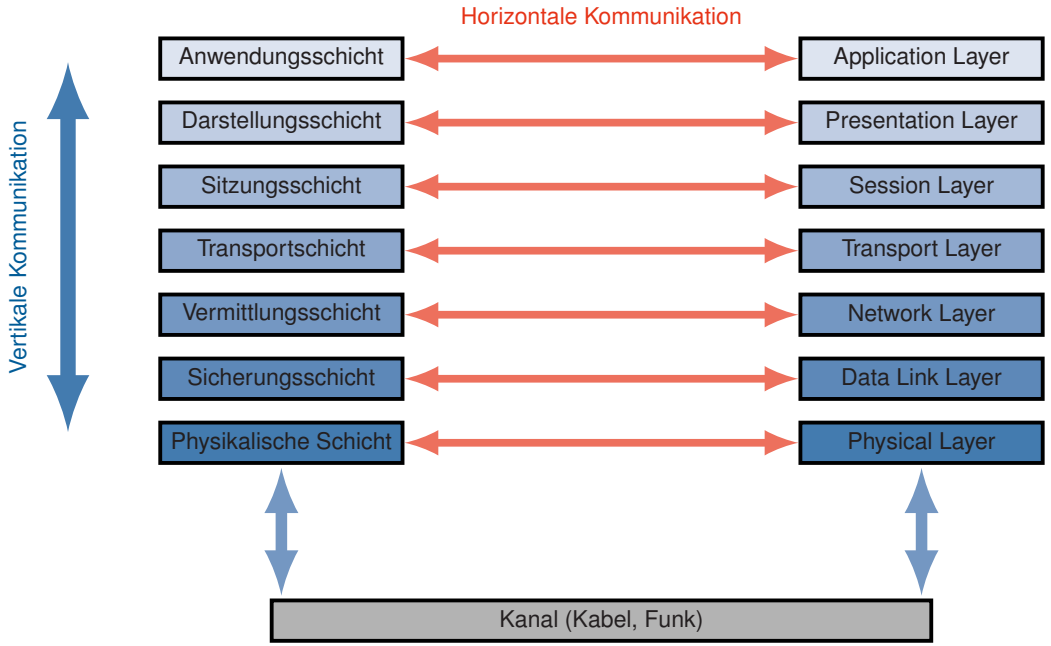
Ablaufbeispiel



Ablaufbeispiel

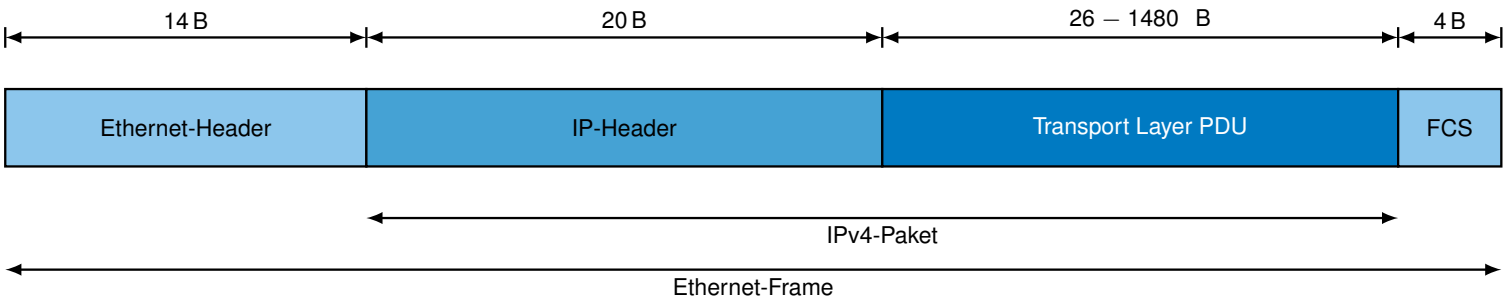


Zusammenfassung: ISO/OSI Modell



- Das ISO/OSI Modell unterteilt den Kommunikationsvorgang in 7 Schichten
- Es spezifiziert, welche Dienste in den verschiedenen Schichten zu erbringen sind
- Es wird jedoch keine Implementation vorgegeben
- Die n-te Schicht des Quellsystems kommuniziert nie direkt mit der n-ten Schicht des Ziels
- Kommunikation erfolgt stets erst vertikal im Stack, dann horizontal im Kanal und abschließend wieder vertikal
- Das Internet ist über einen TCP/IP-Stack realisiert, welcher weniger Schichten umfasst

Zusammenfassung: ISO/OSI Modell



- Den Nutzdaten wird auf jeder Ebene ein entsprechender Header vorangestellt
- Die tatsächlich transportierten Daten beinhalten damit für jede aktive Schicht einen Header

Zusammenfassung: Schichten

Physikalische Schicht
Der von der ersten Schicht angebotene Dienst ist die Punkt-zu-Punkt-Übermittlung von reinen Bits in Form von physikalisch messbaren Signalen.

- Generierung von Signales aus Bits (Impulsformung), welche auf Zielseite wiedererkannt werden müssen (Detektion)
- Auftragen der Signale auf eine Trägerwelle (Modulation)
- Als Trägermedium werden i. d. R. elektromagnetische Wellen genutzt
- Teilweise ausgleichen der inhärenten Unzuverlässigkeit des Kanals mithilfe einer Kanalkodierung

Sicherungsschicht
Die Sicherungsschicht übernimmt die Abstraktion eines physischen Kanals auf eine logische Direktverbindung.

- Erkennen von Übertragungsfehlern, nach Möglichkeit Korrektur
- Verhindern einer Überforderung des Zielsystems bei der Kommunikation (Flusskontrolle)
- Regelung des Medienzugriffs auf ein gemeinsam genutztes Kommunikationssystem

Zusammenfassung: Schichten

Vermittlungsschicht
Die Vermittlungsschicht verbindet Systeme über beliebig viele Direktverbindungen hinweg.

- Ermöglichung von Kommunikation über Direktverbindungen und Subnetze
- Bereitstellung einer eindeutigen und logischen Adressierung von Hosts
- Bestimmung möglichst optimaler Vermittlungspfade zwischen kommunizierenden Hosts (Routing)

Transportschicht
Die Transportschicht realisiert eine Ende-zu-Ende-Kommunikation zwischen Prozessen auf unterschiedlichen Hosts.

- Flusskontrolle zur Verhinderung von Überlast beim Empfänger
- Staukontrolle zur Vermeidung von Überlastsituationen im Netz
- Multiplexing von Datenströmen verschiedener Anwendungen bzw. ihrer Instanzen
- Bereitstellung verbindungsloser bzw. -orientierter Transportmechanismen

Zusammenfassung: Schichten

Sitzungsschicht
Die Sitzungsschicht erlaubt die Etablierung eines gemeinsamen Kommunikationszustandes, der mehrere Verbindungen auf der Transportschicht umfassen kann.

- Stellt Synchronisationspunkte für die Kommunikation zur Verfügung
- Ermöglicht damit das Suspendieren und Wiederaufnehmen von Kommunikationen
- Realisiert einen Koordinationsmechanismus für Kommunikationspartner

Darstellungsschicht
Die Darstellungsschicht ermöglicht die Bereitstellung eines abstrakten Formats zur Repräsentation der übertragenen Daten.

- Abstraktion von anwendungsspezifischer Syntax
- Bereitstellung von Datenkompression
- Ver- und Entschlüsselung der Kommunikationsdaten
- Umkodierung der Kommunikationsdaten

Zusammenfassung: Schichten

Anwendungsschicht
Die Anwendungsschicht beinhaltet alle Protokolle, welche direkt mit Anwendungen interagieren und deren Datentransport realisieren.

- Stellen einen Dienst für den User zur Verfügung
- Besitzen kein gemeinsames Dienste-Interface, da ihr Einsatzzweck sehr unterschiedlich ist

Die durch das ISO/OSI Modell realisierten Abstraktionen ermöglichen es Prozessen auf verschiedenen Systemen, transparent über ein Netz zu kommunizieren, ohne sich mit dem eigentlichen Übermittlungsvorgang auseinandersetzen zu müssen.
Die Schichtarchitektur selbst erlaubt, dass die Technologien einzelner Segmente ausgetauscht werden können, ohne dass Änderungen am restlichen Stack vorgenommen werden müssen (z. B. kabelgebundene Kommunikation vs. WLAN).
Zusätzlich ist es möglich, Adapter zur Verfügung zu stellen, welche die gleichzeitige Nutzung verschiedener Technologien während ein und demselben Datenaustausch ermöglichen (z. B. WLAN Access Point).