

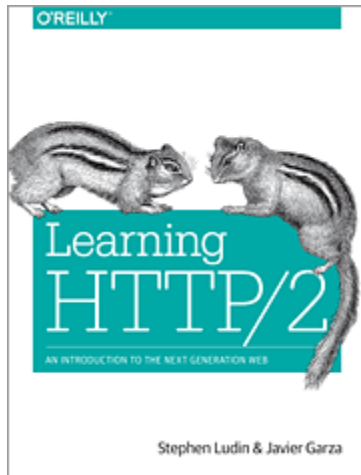
Webtechnologien

Prof. Dr.-Ing. Georg J. Schneider

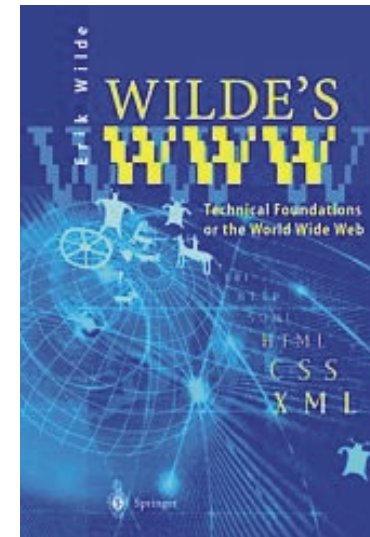
Multimedia und Medieninformatik
Fachbereich Informatik
Hochschule Trier

Literatur

Learning HTTP/2 O'Reilly



Wilde's WWW, Technical Foundations of the World Wide Web Erik Wilde, Springer



Internetworking: Technische Grundlagen und Anwendungen Christoph Meinel, Harald Sack, Springer

Elemente des Web

- Identifikationsschema
 - Uniform Resource Identifier, URI
 - Verwendung als Uniform Resource Name, URN
 - Verwendung als Uniform Resource Locator, URL
- Transferprotokoll
 - Hypertext Transfer Protocol, HTTP
 - ASCII-kodiertes Request-Reply Protokoll über TCP/IP
- Dokumentformat
 - Hypertext Markup Language, HTML
 - Abstrakte Sprachspezifikation mit mehreren Ausprägungen

Transferprotokoll

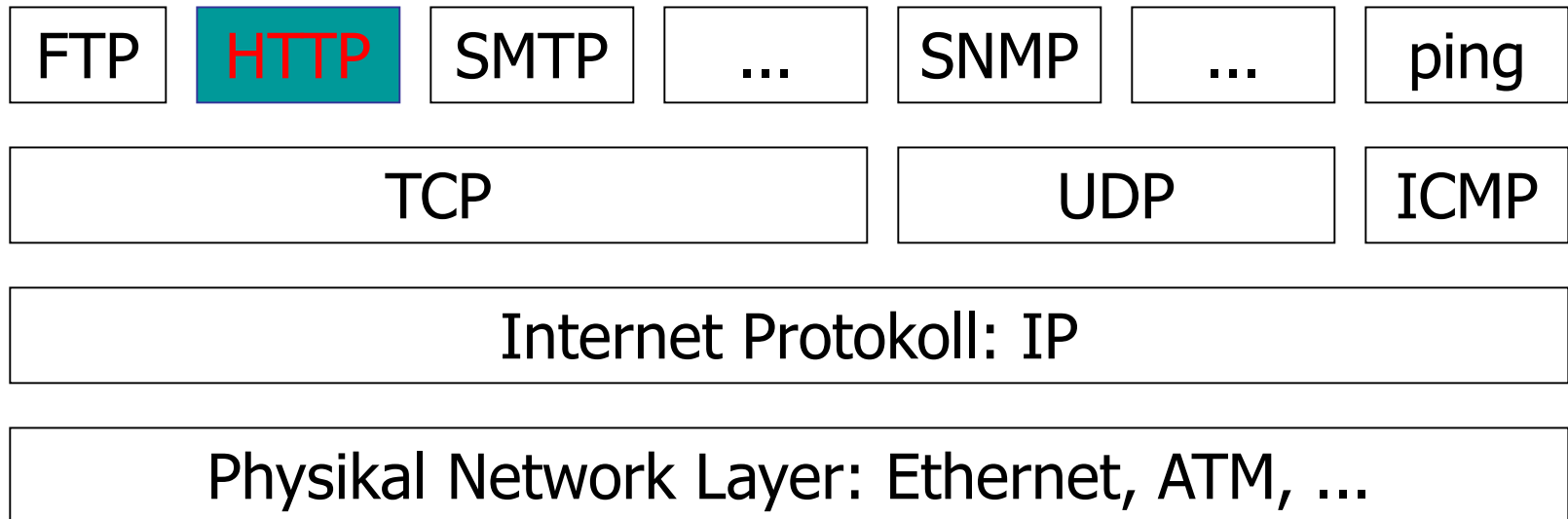
- Identifizierte Ressourcen müssen transportiert werden
- Client-Server Architektur fordert
 - Request-Reply Protokoll
 - Transaktionscharakter
- Entwurfsziele
 - einfach und leichtgewichtig
 - schnell
- Hypertext Transfer Protocol, HTTP
 - basierend auf TCP/IP
 - idempotent, zustandslos
 - ASCII kodiert

Warum in der Vorlesung?

Wichtig für:

- Web Server Administratoren
- Programmierer von Web Anwendungen
- Content Provider

Internet Protocol Suite



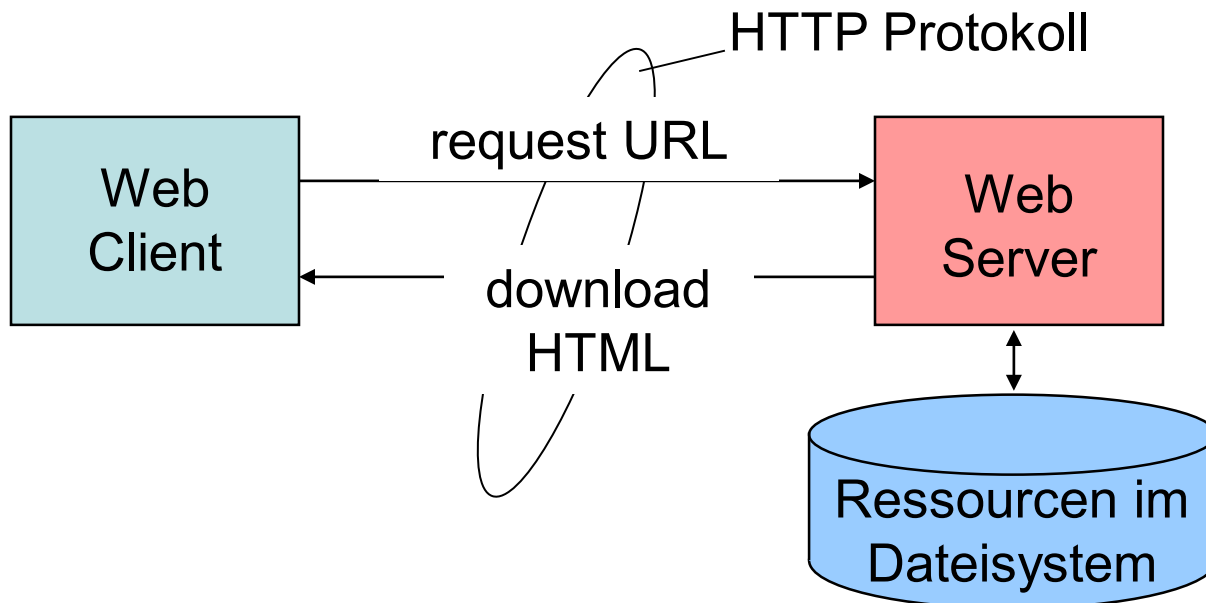
Grundsatz zum Parsen / Generieren von HTTP Nachrichten

Tolerant mit den Daten umgehen,
die ankommen.

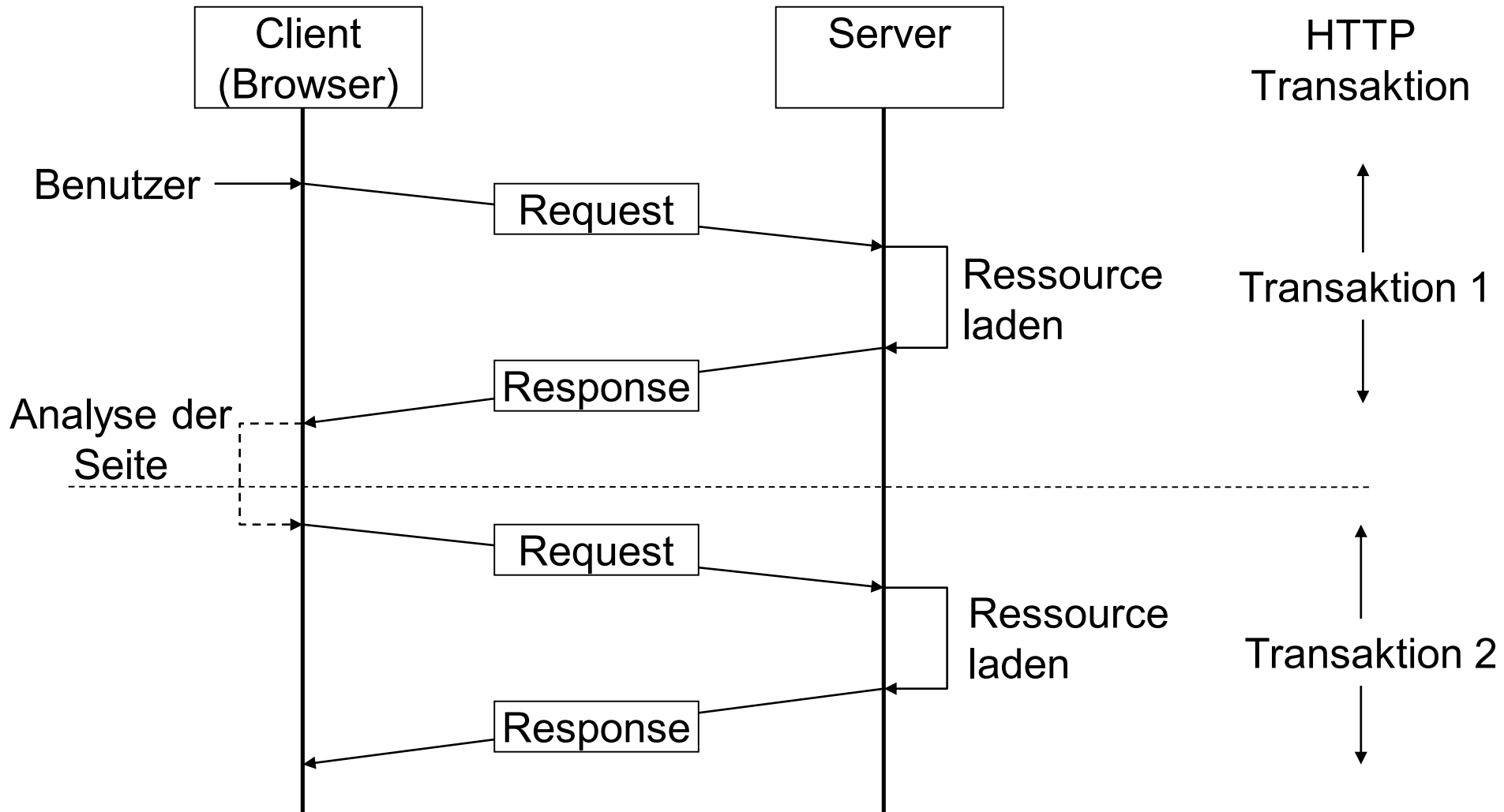
Konservativ mit den Daten umgehen,
die versendet werden.

Architektur und Protokolle

- Client-Server Architektur
- synchrones Kommunikationsmodell (Request/Response)
- Ressourcen
 - Einheit der Kommunikation zwischen Client und Server
 - statisch oder dynamisch



HTTP Anfrage



HTTP/1.0

- Nur ein informational RFC, 1992-1996 [RFC1945]
- Message Types
 - Request (GET, HEAD, POST)
 - Response
- Header Fields
 - variable Anzahl
 - Syntax: <field_name> ":" <field_value>
 - Transfer von Metainformation zu Request, Response, Inhalt

HTTP/1.0 contd.

- Response Codes
 - Status- und Fehlermeldungen
- Media Types
 - Transfer beliebiger Nichttext-Ressourcen
 - insbesondere Graphik, Bilder, Audio, Video
 - basierend auf MIME (Multipurpose Internet Mail Extensions)
- Basic Authentication

HTTP/1.0 contd.

- MIME Types [RFC2045, 2046]

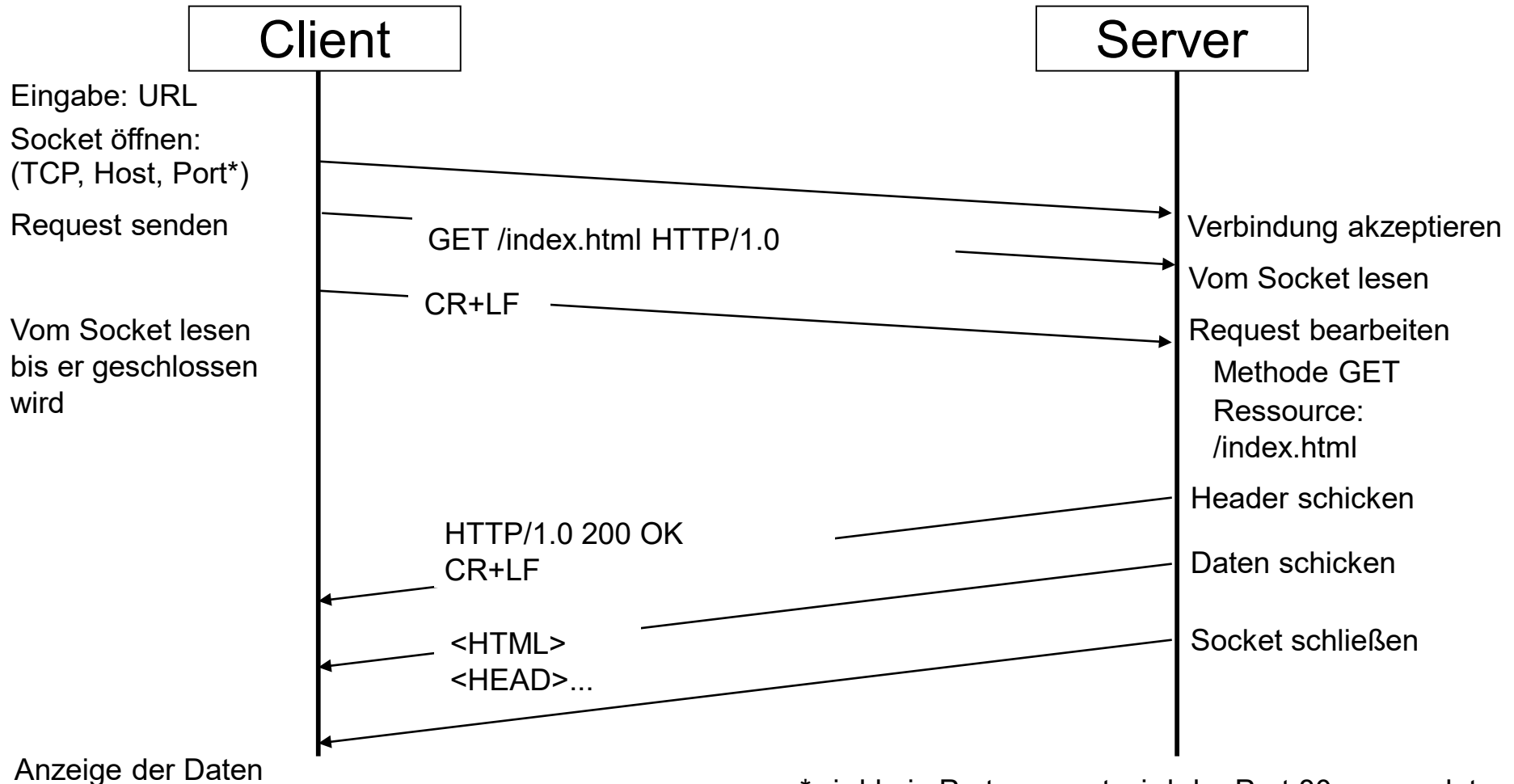
Aufbau (vereinfacht):

- **type "/" subtype**

Beispiel

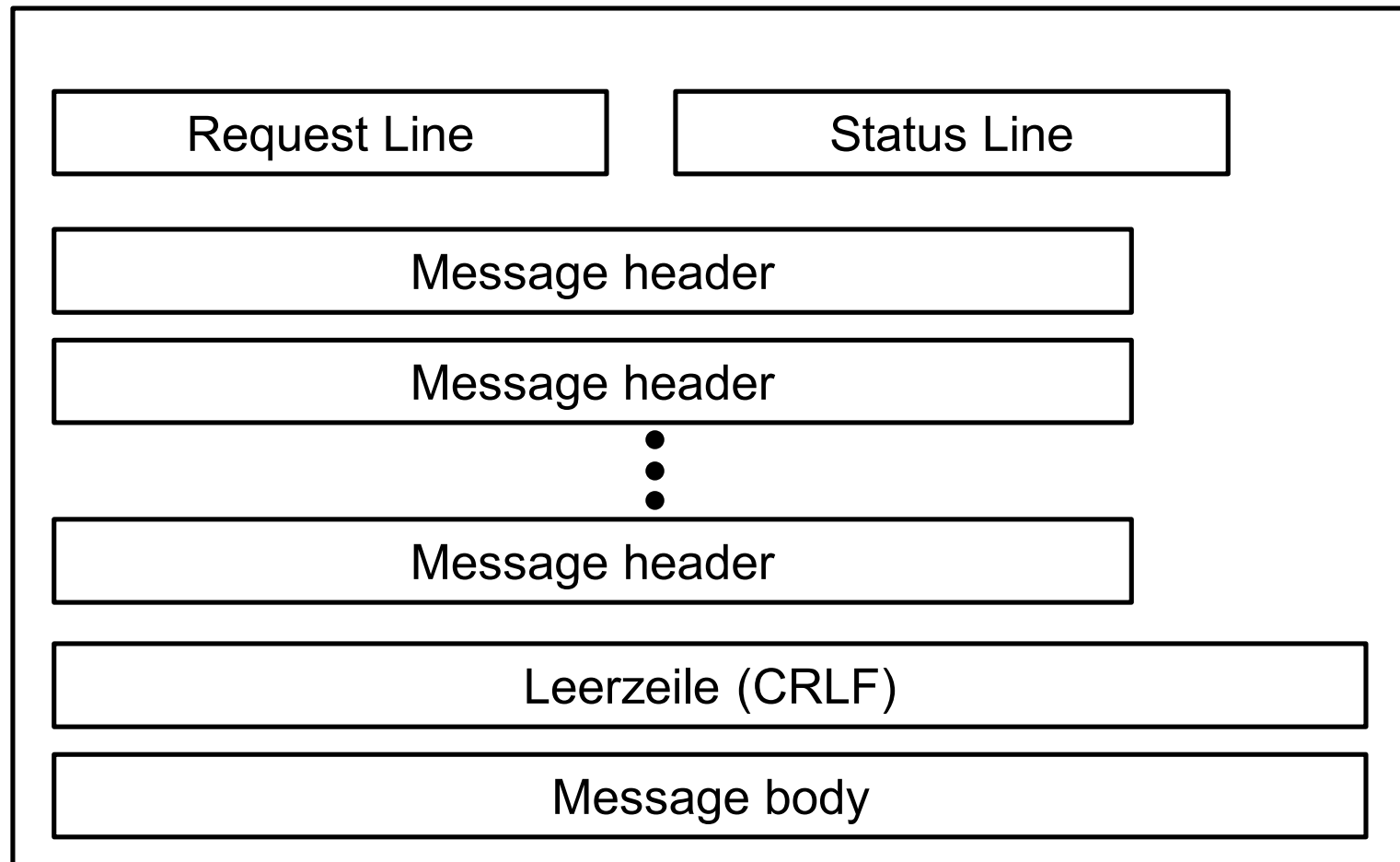
- Bitmap Image File
- MIME Type: **image/bmp**
- Dateiendung **.bmp**

Dokumententransfer (HTTP/1.0)



*wird kein Port genannt wird der Port 80 verwendet

HTTP Nachricht



HTTP/1.0 Beispiel I

Request

GET /index.html HTTP/1.0

Accept: image/gif, image/x-xbitmap, image/jpeg, */*

Accept-Language: de

Accept-Encoding: gzip

User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT)

Request Methode

Header

Leerzeile

HTTP/1.0 Beispiel II

Response

The diagram shows an HTTP response structure with labels on the right and arrows pointing to the corresponding parts of the response text on the left. The response text is as follows:

```
HTTP/1.0 200 OK
Server: ServerName
Content-Type: text/html
Content-Length: 80

<HTML>
<TITLE> ...
```

The labels and their corresponding parts are:

- Status**: Points to the first line, `HTTP/1.0 200 OK`.
- Header**: A bracket groups the next three lines: `Server: ServerName`, `Content-Type: text/html`, and `Content-Length: 80`.
- Leerzeile**: Points to the empty line separating the header from the body.
- Payload**: Points to the body content, starting with `<HTML>`.

Dokumente beinhalten Ressourcen I

Antwort des Servers

```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 3213
```

```
<html>
<head>
<title>Oracle Corporation - Home</title>
...
</head>
<body bgcolor="#ffffff" link="#000000" vlink="#ff0000">
...
<INPUT NAME=q size=10 maxlength=800 VALUE=""><INPUT
TYPE="image" src="/templates/images/search_btn.gif"
width=36 height=18 value="go" border=0>
...
<a href="/html/dev_it.html">
</a>
...

...
</body>
</html>
```

Dokumente beinhalten Ressourcen II

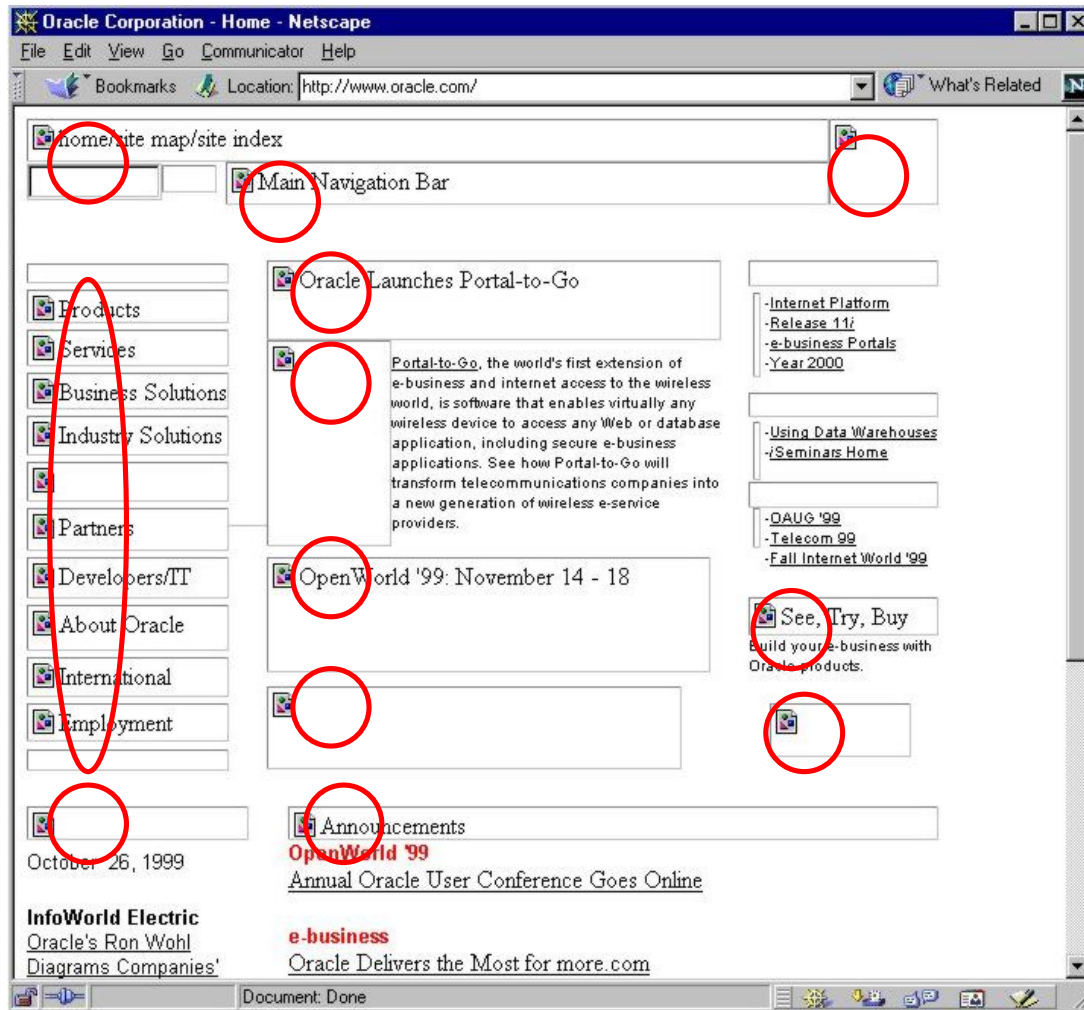
Antwort des Servers

```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 3213

<html>
<head>
<title>Oracle Corporation - Home</title>
...
</head>
<body bgcolor="#ffffff" link="#000000" vlink="#ff0000">
...
<INPUT NAME=q size=10 maxlength=800 VALUE=""><INPUT
TYPE="image" src="/templates/images/search_btn.gif"
width=36 height=18 value="go" border=0>
...
<a href="/html/dev_it.html">
</a>
...

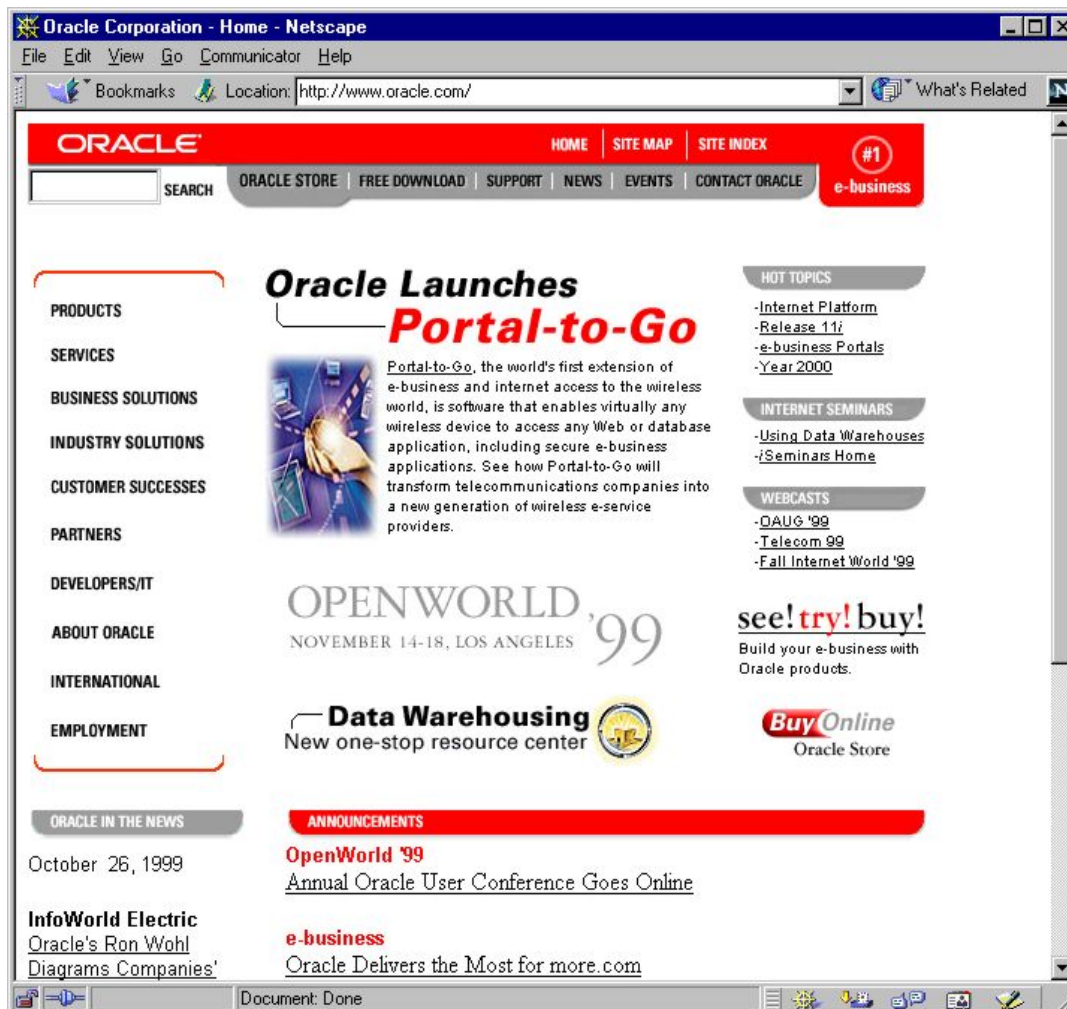
...
</body>
</html>
```

Dokumente beinhalten Ressourcen III



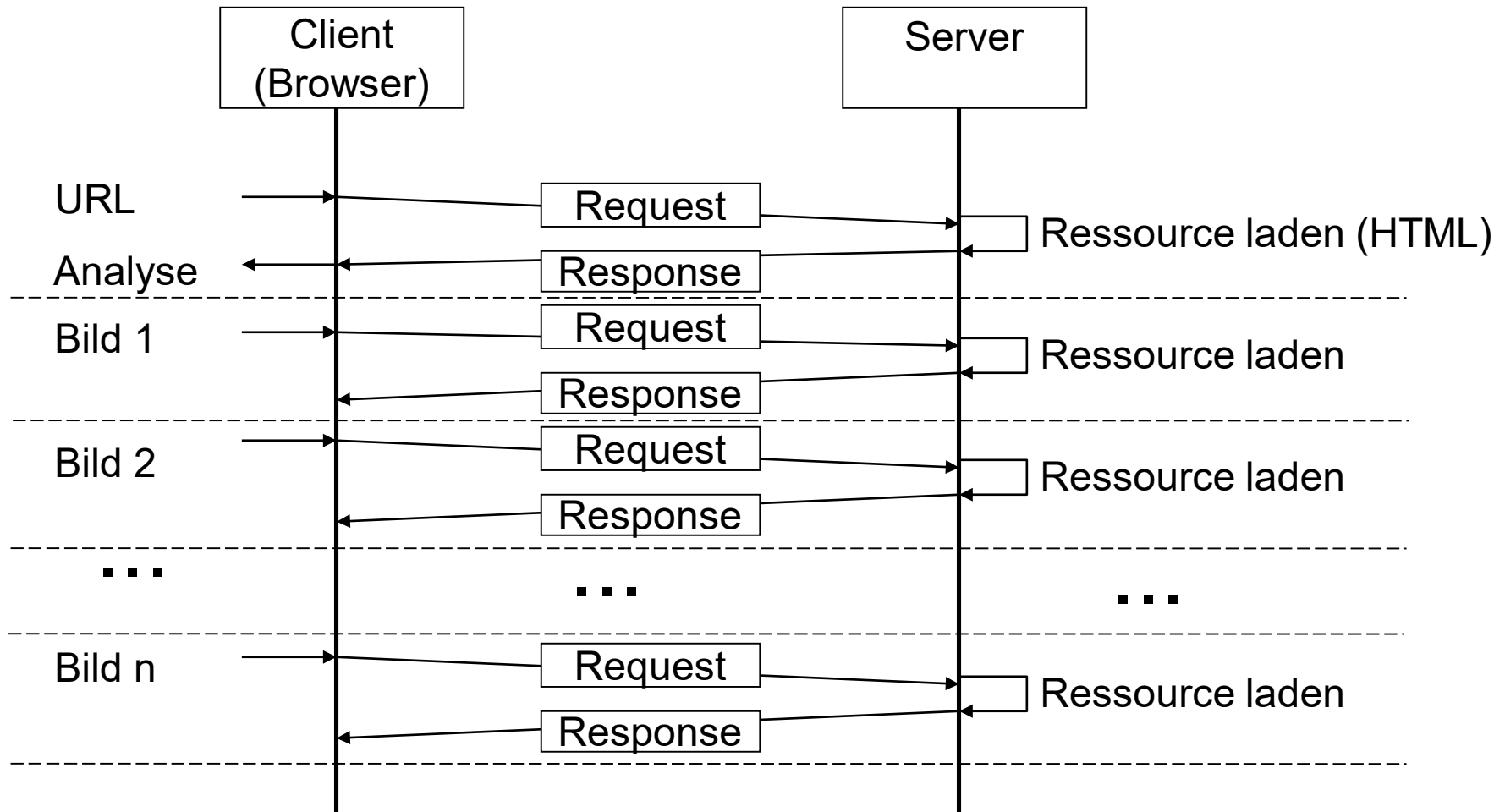
- Bilder
- Hintergrund
- Schaltflächen
- Musik
- Geräusche

Dokumente beinhalten Ressourcen IV



- Bilder
- Hintergrund
- Schaltflächen
- Musik
- Geräusche

Dokumente beinhalten Ressourcen V



HTTP Versionen

Schwächen

Eine TCP Verbindung pro HTTP Transaktion

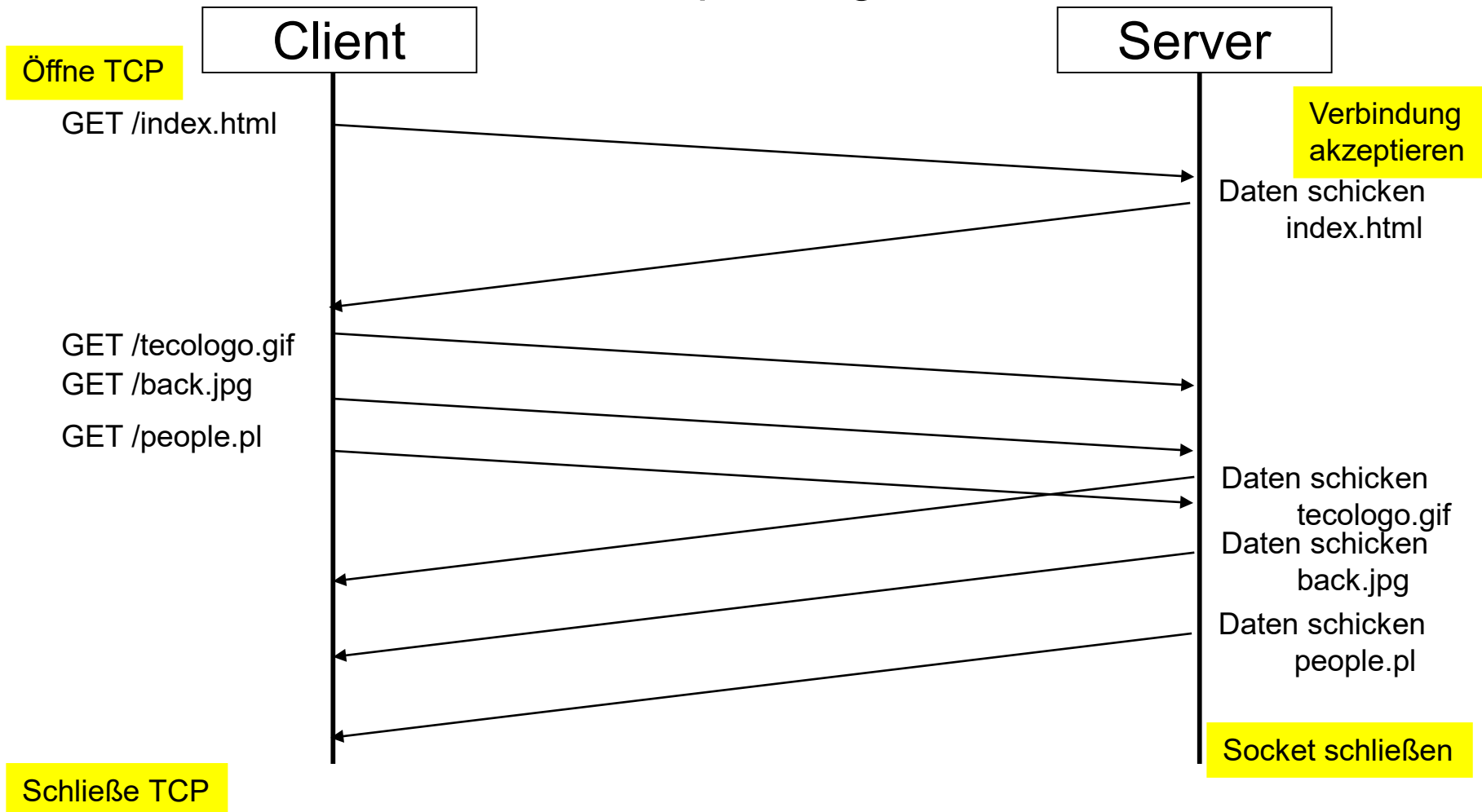
- HTTP/1.1: mehrere Requests während einer TCP Verbindung
- HTTP/2 Request and Response Multiplexing

Öffne TCP



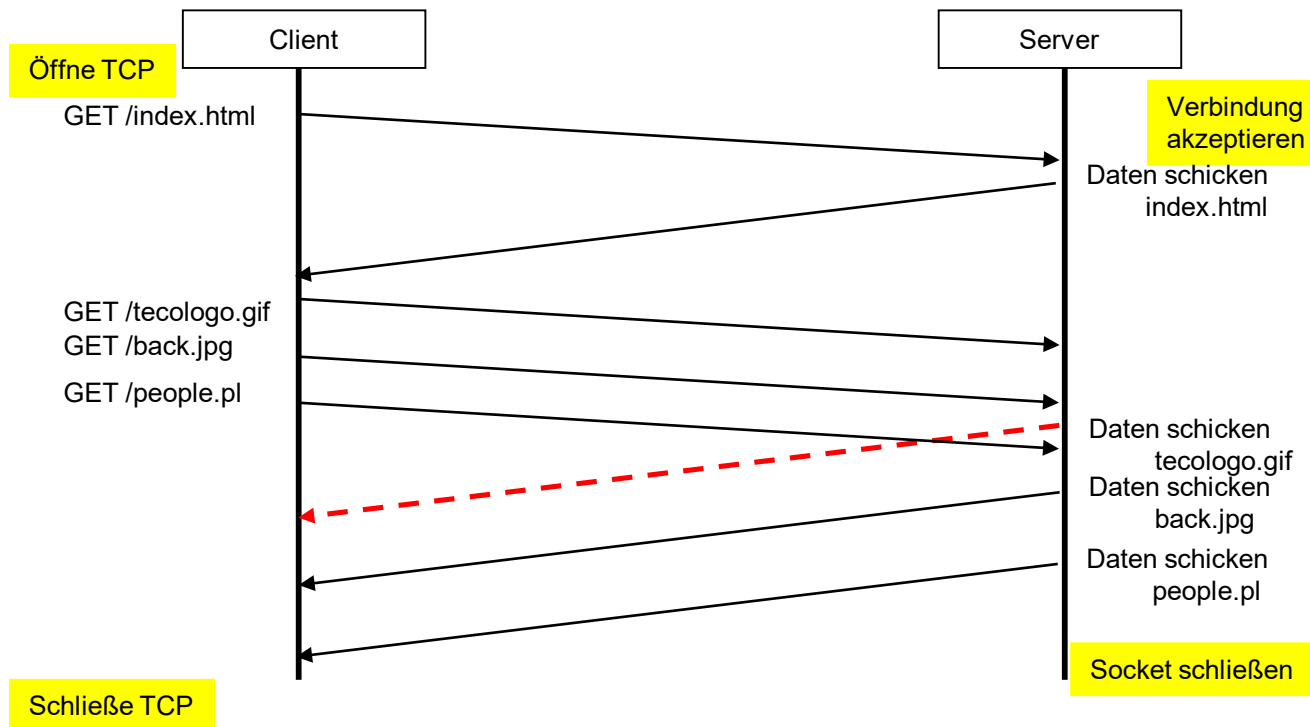
Persistente Verbindungen in HTTP/1.1

Pipelining



HTTP/1.x Probleme

- Head of Line blocking (HOL)



HTTP/2 Server Push

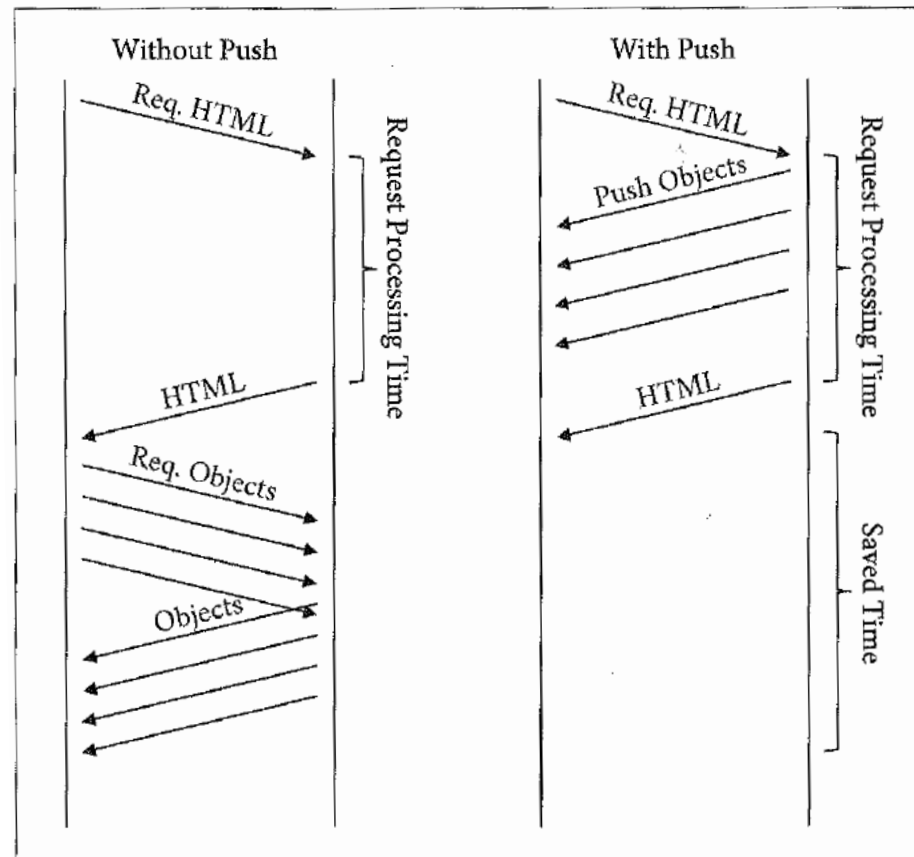
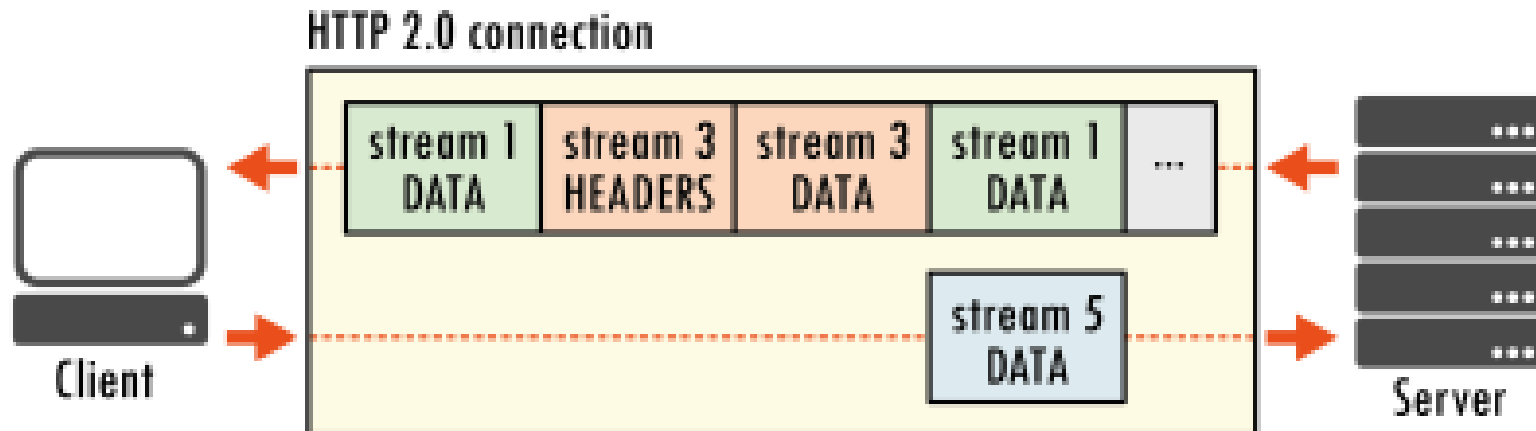


Figure 6-5. Push while processing

HTTP/2 Streams

Request and Response Multiplexing



HTTP/1.0

Basic Authentication I

- Restriktiver Zugang zu Ressourcen
- Protokollieren der Benutzer (Namen) die zugreifen
- Basic Authentication
 - einfaches Username - Password Schema
 - **<user>:<passwd>** Base64 kodiert
 - Keine Verschlüsselung!

HTTP/1.0

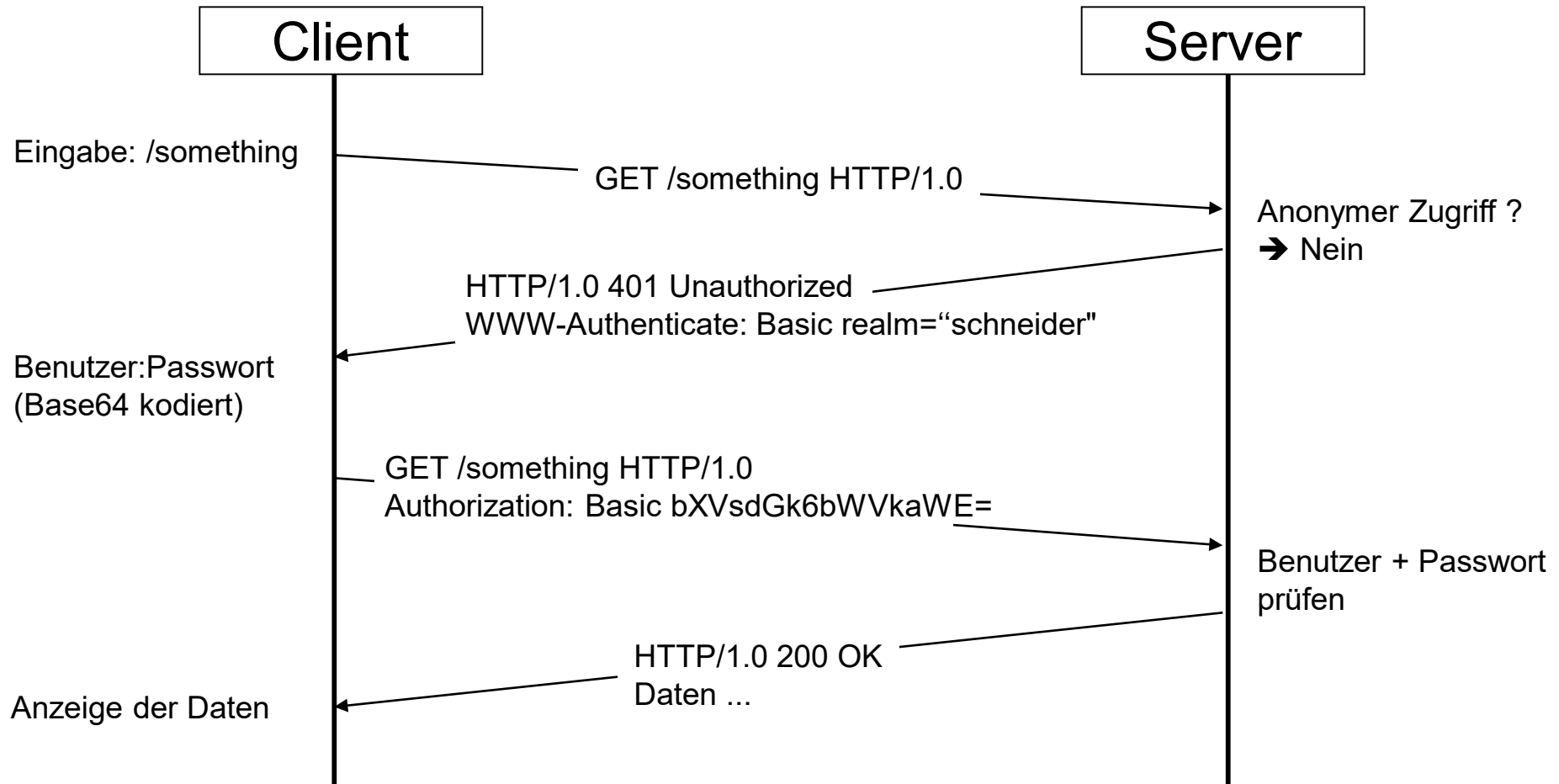
Basic Authentication II

Ablauf

- Anfrage einer Ressource
- Antwort mit Status-code: **401 Unauthorized**
und Header: **WWW-Authenticate**
- Anfrage einer Ressource mit zusätzlichem Header
Authorization: <user>:<passwd> (Base64 kodiert)
- Überprüfen von **<user>:<passwd>** beim Server
- Rückgabe des Dokuments

HTTP/1.0

Basic Authentication III



HTTP/1.0

Schwäche

Basic Authentication überträgt Benutzername und Passwort unverschlüsselt über das Internet

- Zusätzliche in HTTP/1.1: Digest Access Authentication
 - Keine Übermittlung des Passworts
 - Problem: initiale Übermittlung des Passworts

HTTP/1.0

Schwäche

Keine Unterstützung für non-IP-based virtual Hosts
mehrere Web Server laufen auf einer Maschine mit einer IP-Adresse

- In HTTP/1.1: Unterstützung von “non IP based Virtual Hosts” unter Verwendung des Header Field: **HOST**

Non-IP-based Virtual Hosts

- HTTP-Requests verwenden das Headerfeld **Host: <hostname>**
und/oder den absoluten URI
- HTTP/1.1 Anfrage ohne Hostname resultiert in einem Fehler, z.B. folgender Errorlog-Eintrag
`[access to /index.html failed for 129.13.170.1,
reason: client sent HTTP/1.1 request without
hostname]`
- Alle Domains auf einem Server können auf die gleiche IP-Adresse und den gleichen Port abgebildet werden
- Die Domain wird durch die Angabe des Hosts im Header-Feld **HOST** und nicht durch den Aufbau der TCP-Verbindung ausgewählt

HTTP/1.0

Schwäche

Primitives Caching Model

- Fehlende Unterstützung für Proxies und Gateways

HTTP/1.1 – Intermediaries



HTTP/1.1

Intermediaries

- Proxy

Ein Proxy empfängt die an einen Server gerichtete Nachricht, schreibt diese um und schickt sie dann an den Server weiter. Er implementiert Client und Server Part

- Gateway

Ein Gateway dient dazu ein Protokoll für den Server zu übersetzen. Der Server fragt hierbei das direkt Gateway an. Das Gateway sendet darauf beispielsweise der Request an einen weiteren Server um eine Ressource anzufragen.

- Tunnel

Ein Tunnel ist dafür zuständig, eine Nachricht an einer Vermittlungsstelle vorbei unverändert weiterzuleiten.

Er wird nicht als Teil der HTTP-Kommunikation angesehen.

Caching / Proxies

Cache

- Enthält lokale Kopien von Antworten auf Anfragen. Hierdurch soll das Netzwerk weniger belastet und die Geschwindigkeit erhöht werden. Die Cache-Verwaltung kümmert sich um das Speichern, Versenden und Löschen dieser Antworten.

Unterstützung von Caching

- Header Fields wie z.B.:
Cache-Control, Expires, Age, Pragma

Caching-Proxy

Beispiel MISS

Cache – MISS

- Die angefragte Ressource befindet sich nicht im Cache
- die Ressource wird beim Originalserver nachgefragt



Cache-MISS - Anfrage über Proxy

Beispielantwort

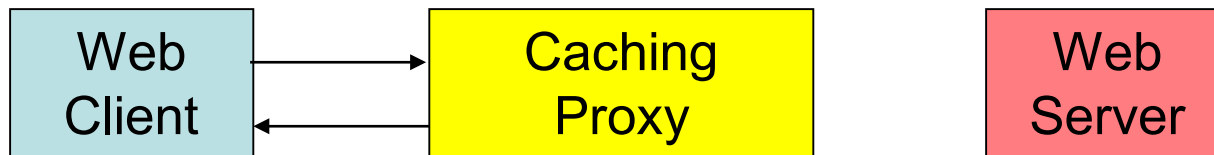
```
HTTP/1.0 200 OK
Date: Fri, 21 Sep 2001 14:52:18 GMT
Server: Apache/1.3.19 (Unix) ApacheJServ/1.1.2 PHP/4.0.4p11
       mod_ssl/2.8.1 OpenSSL
L/0.9.6a
Cache-Control: max-age=315019
Expires: Tue, 25 Sep 2001 06:22:37 GMT
Last-Modified: Tue, 18 Sep 2001 06:22:37 GMT
ETag: "1cbf13f-27a4-3ba6e82d"
Accept-Ranges: bytes
Content-Length: 10148
Content-Type: text/html
X-Cache: MISS from www-cache.fh-trier.de
Proxy-Connection: keep-alive
...
```

Caching-Proxy

Beispiel HIT

Cache – HIT

- Die angefragte Ressource befindet sich im Speicher des Caching-Proxies und ist noch gültig
- die Ressource wird aus dem Cache zurückgeschickt



Cache-HIT - Anfrage über Proxy

Beispielantwort

```
HTTP/1.0 200 OK
Date: Fri, 21 Sep 2001 14:52:18 GMT
Server: Apache/1.3.19 (Unix) ApacheJServ/1.1.2 PHP/4.0.4p11
       mod_ssl/2.8.1 OpenSSL/0.9.6a
Cache-Control: max-age=315019
Expires: Tue, 25 Sep 2001 06:22:37 GMT
Last-Modified: Tue, 18 Sep 2001 06:22:37 GMT
ETag: "1cbf13f-27a4-3ba6e82d"
Accept-Ranges: bytes
Content-Length: 10148
Content-Type: text/html
Age: 194
X-Cache: HIT from www-cache.fh-trier.de
Proxy-Connection: keep-alive
...
```


HTTP

Methoden

GET	Anfragen einer Ressource
POST	Anlegen einer Ressource (oder ändern)
HEAD	Nur Header, nicht Body senden
PUT	Ersetzt eine Ressource (oder legt sie an)
DELETE	Löschen einer Ressource
TRACE	Liefert den Transportweg bzgl. HTTP
OPTIONS	Liste der unterstützten Methoden
CONNECT	Stellt einen Tunnel zur Verfügung und leitet weiter

HTTP

Methoden

Am meisten verwendet: GET, POST

- GET: Anfragedaten in der URI
<https://www.google.com/search?client=firefox-b-d&q=hamburger+rezept>
 - Loggen der Daten auf dem Server
 - Bleiben in der History
 - Größenbeschränkung

- POST: Anfragen im Message Bod

```
POST / HTTP/1.1
User-Agent: Java/1.4.2_01
Host: 127.0.0.1:88
Accept: text/html;
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
```

isVorname=P%C3%B6%C3%9Ftel

Aufbau von HTTP-Nachrichten

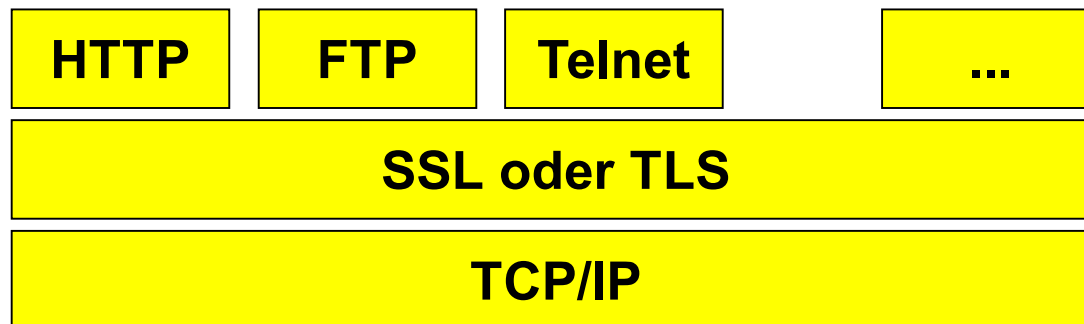
Code	Gruppe	Bedeutung
1xx	Informational	Bearbeitung der Anfrage dauert noch an
2xx	Success	Anfrage war erfolgreich
3xx	Redirection	Bearbeitung erfordert weitere Schritte des Clients
4xx	Client Error	Ursache des Scheiterns liegt bei Client
5xx	Server Error	Ursache des Scheiterns liegt bei Server

Response Status Codes (Auszug)


```
Status-Code = "100" ; Continue
              | "101" ; Switching Protocols
              | "200" ; OK
              | "201" ; Created
              | "202" ; Accepted
              | "203" ; Non-Authoritative Information
              | "204" ; No Content
              | "300" ; Multiple Choices
              | "301" ; Moved Permanently
              | "400" ; Bad Request
              | "401" ; Unauthorized
              | "402" ; Payment Required
              | "403" ; Forbidden
              | "404" ; Not Found
              | "405" ; Method Not Allowed
              | "500" ; Internal Server Error
              | "501" ; Not Implemented
```

Hypertext Transfer Protocol Secure HTTPS

HTTPS = HTTP über SSL/TLS
(Secure Socket Layer, Transport Layer Security)



 Sparkassen-Finanzportal GmbH [DE] | <https://www.sparkasse.de/>

 Sparkassen-Finanzportal GmbH [DE] | www.sparkasse.de

 Sparkassen-Finanzportal GmbH [DE] | [sparkasse.de](https://www.sparkasse.de/)

  Sparkassen-Finanzportal GmbH (DE) | https://www.sparkasse.de

Content Negotiation

- sprachspezifische Ressourcen
- Ressourcen unterschiedlicher Qualität
- Ressourcen unterschiedlicher Kodierung
- Header Fields wie z.B.:
 - **Accept,**
Accept-Charset,
Accept-Language,
Accept-Encoding, ...

Conditional GET

Syntax

**GET <URI> <VERSION>
<CONDITIONAL-HEADER>: <DATE>**

z.B. If-Modified-Since, If-Match, If-Range, etc.

Conditional GET

Beispielanfrage

```
GET http://www.apache.org/index.html HTTP/1.1
Host: www.apache.org
If-Modified-Since: Fri, 29 Oct 1999 13:53:40 GMT
```


Conditional GET

Beispielantwort

HTTP/1.0 304 Not Modified

Date: Thu, 28 Oct 1999 13:55:13 GMT

Content-Type: text/html

Expires: Fri, 29 Oct 1999 13:55:13 GMT

Weitere Headerfelder

Referer

- Enthält die Informationen über die URL der zuletzt besuchte Seite
- Wird i.a. vom Browser nur mitgeschickt wenn die URL durch Anklicken eines Links angewählt wurde
- Verwendung:
 - Sicherstellen, dass eine Funktion bzw. eine Seite nur von einer bestimmten vorgegebenen Adresse aus aufgerufen wird
 - Nachvollziehen woher Benutzer auf die eigene Seite kommen
z.B. Feststellen welche Werbung sich lohnt
- z.B. **Referer:** `http://www.fh-trier.de/index.html`

Server-initiierte Kommunikation

HTTP ist ein Request-Reply Protokoll

- Anfragen gehen immer vom Client aus (Änderung bei HTTP/2)
- Server kann keine Kommunikation zum Client aufbauen

Abhilfe: Client muss die Information periodisch nachfragen

- Problem: Skalierbarkeit
- Standard-Browser

alle 60 Sekunden Seite erneut laden:

```
<META HTTP-EQUIV="Refresh" content="60">
```

umleiten auf URL:

```
<META HTTP-EQUIV="Refresh" content="0;  
URL=http://www.fh-trier.de/">
```

- Spezielle Clients

Cookies

HTTP ist zustandslos

- es besteht kein Zusammenhang zwischen zwei Anfragen

Cookies speichern den Zustand beim Client

- RFC6265

Protokollprimitive

- `Set-Cookie` wird vom Server zum Client geschickt
- `Cookie` wird vom Client zum Server geschickt
abhängig von der Domain und dem aktuellen Pfad

Cookies

Beispiel Setzen des Cookie

```
HTTP/1.1 200 OK
Date: Sat, 21 Oct 2017 20:45:52 GMT
Server: Server
Set-Cookie: session-id=260-4262357-9809340; Domain=.amazon.de;
Expires=Tue, 01-Jan-2036 08:00:01 GMT; Path=/
Connection: Keep-Alive
Content-Language: de-DE
Content-Type: text/html; charset=UTF-8

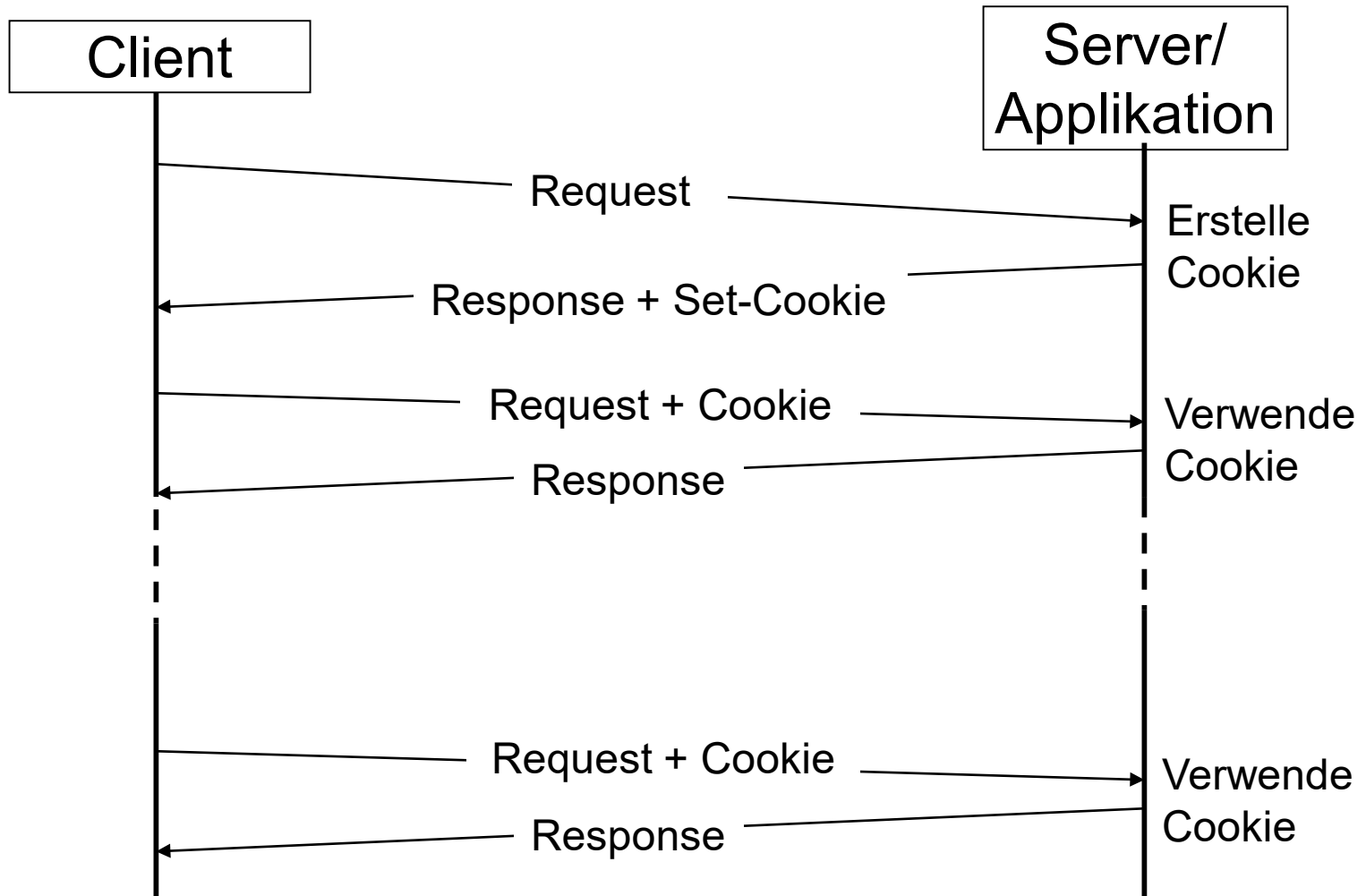
<html>
...
```

Cookies

Beispielanfrage bei vorhandenem Cookie

```
GET / HTTP/1.1
Accept: */*
Connection: Keep-Alive
Cookie: ad-id=AyaxcvlTTUO4rUIkbCOds1Q; ad-privacy=0
Host: aax-eu.amazon-adsystem.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
```

Zustandsbehaftete Kommunikation



Ausblick

HTTP/3

- <https://quicwg.org/base-drafts/draft-ietf-quic-http.html>
- HTTP/3 ist die dritte Version des Hypertext Transfer Protocol (HTTP)
- Früher: HTTP-over-QUIC. QUIC (Quick UDP Internet Connections)
- Ursprünglich von Google entwickelt
- Nachfolger von HTTP/2 (Verwendung von HTTP/2 aktuell ca. 30%)
- Noch nicht fertig spezifiziert, Hypertext Transfer Protocol Version 3 (HTTP/3) draft-ietf-quic-http-32 vom 20 Oktober 2020, Expires: 23 April 2021
- Wird bereits von Google im Internet verwendet (Google Chrome, YouTube, Gmail, Google Suche)