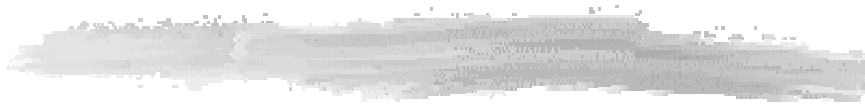


# WWW - Grundlagen und Technologie

## Hypertext Transfer Protocol (HTTP)



**Erik Wilde**  
**TIK - ETH Zürich**  
**Sommersemester 2001**

## Übersicht



- Hypertext braucht Ressourcen-Identifikation
  - Unterschied zwischen Link und Identifier
- *Universal Resource Identifier (URI)*
  - *Uniform Resource Locator (URL)*
  - Identifikation von Ressourcen-Fragmenten
- *Hypertext Transfer Protocol (HTTP)*
  - Interaktionen und Message Typen
  - Performance-Fragen und -Optimierungen
  - Caches und Proxies
- *Distributed Authoring and Versioning (WebDAV)*
- *HTTP over SSL (HTTPS) als "sicheres HTTP"*

## Identifizier vs. Links

- *Identifizier*
  - Identifizierung unabhängig von Verbindungen
  - Identifizier existieren auch ohne Referenzen

  
<a href="http://www.ethz.ch/">ETH</a>

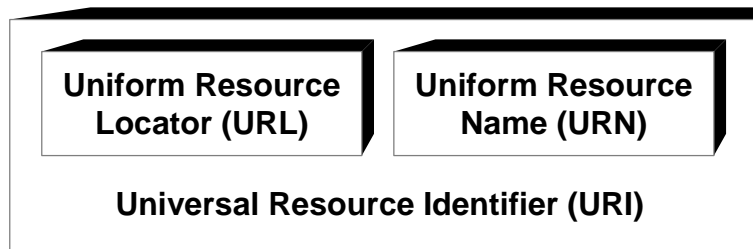
- *Links*
  - enthalten (u.U. mehrere) Identifizier
  - zusätzliche Semantik (z.B. Link-Typ)
  - HTML definiert sehr einfache Links (→ *XLink*)

## Universal Resource Identifier (URI)

uri = scheme ":" scheme-specific-part

- definiert in Internet RFC 2396
- *URI schemes*
  - definieren eine Art der Identifizierung
  - definieren Identifizierung innerhalb des schemes
- *URI scheme specific parts*
  - Bedeutung hängt vom scheme ab
  - einige allgemeine Festlegungen zur Syntax
- URI sind *Locators (URL)* oder *Names (URN)*

## Locators and Names



"The name of a resource indicates what we seek, and an address indicates where it is."

## URI Syntax

- Sonderzeichen
  - werden durch das Prozentzeichen eingeleitet
  - `http://www.ethz.ch/%25-Zeichen`
- Hierarchische Organisation
  - wird durch einen Slash gekennzeichnet
  - `http://www.ethz.ch/info/telefon.html`
- Fragment-Identifizierung
  - durch ein Doppelkreuz gekennzeichnet
  - `http://www.ethz.ch/index.html#address`
- Query Strings
  - mit einem Fragezeichen gekennzeichnet
  - Spaces werden durch ein Pluszeichen dargestellt
  - `http://www.ethz.ch/telefon?erik+wilde`

## Uniform Resource Locator (URL)

---

- definiert in Internet RFCs 1738, 1808, 2368, 2384, und 2397
- erste Festlegung 1989/90
- Anforderungen an Identifier
  - Erweiterbarkeit (neue Zugriffsmethoden)
  - Komplettheit (flexibel genug)
  - Druckbarkeit (Austausch und Niederschrift)
- Unterstützung mehrerer URL schemes
  - Implementierung verschiedener Protokolle
  - Trend zu integrierten Arbeitsumgebungen
  - `ftp://ftp.ethz.ch/pub/hg.jpeg`

## URL schemes

---

- *http* (Hypertext Transfer Protocol)
  - Uebertragungsprotokoll des WWW
- *https* (HTTP over SSL)
  - sichere Variante von HTTP
  - basiert auf Secure Sockets Layer (SSL)
- *mailto* (electronic mail)
  - Mailadresse gemäss RFC 822
  - Versenden mittels SMTP (RFC 821)
- *ftp* (File Transfer Protocol)
  - File-Uebertragung gemäss RFC 959
- *news* (Usenet news)
  - Newsgroups oder Artikel (RFC 977)

## URL scheme specific part

---

```
"//" [ user [ ":" password ] "@" ]  
    host [ ":" port ] "/" url-path
```

- spezifiziert Details für ein URL scheme
- optionale Komponenten
  - *user* für eine Benutzerkennung
  - *password* für ein Passwort
  - *port* für einen vom Standard abweichenden Port
- notwendige Komponenten
  - *host* als *fully qualified host name (FQHN)*
  - *url-path* als Adressierung auf dem Host

## Nachteile von URLs

---

- Abhängigkeit vom Rechner
- Abhängigkeit vom Namen des Rechners
- Abhängigkeit vom Server-Layout
- verbreitete Gründe für ungültige URLs
  - Änderung des Namens auf dem Server
  - Änderung des Server-Namens
  - Verschiebung auf einen anderen Server
- naheliegende Lösung: Naming Service
  - eine zusätzliche Indirektion
  - Unabhängigkeit von Adressen (URLs)

## Fragment-Identifizierung

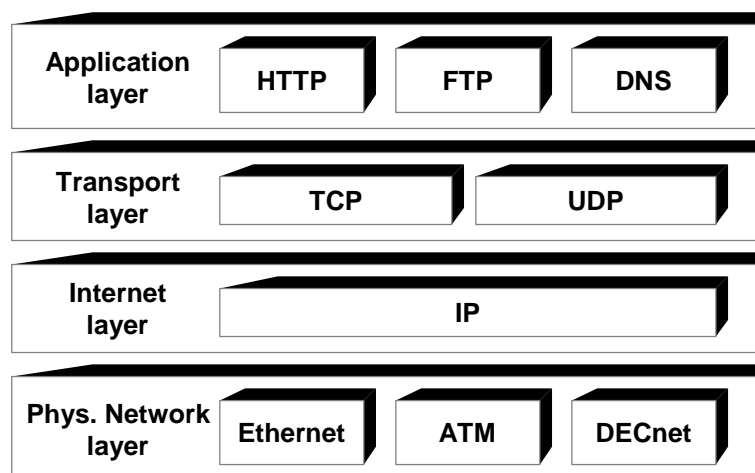
- HTML erlaubt nur explizite Fragmente

URL: `http://www.ethz.ch/index.html#telefon`

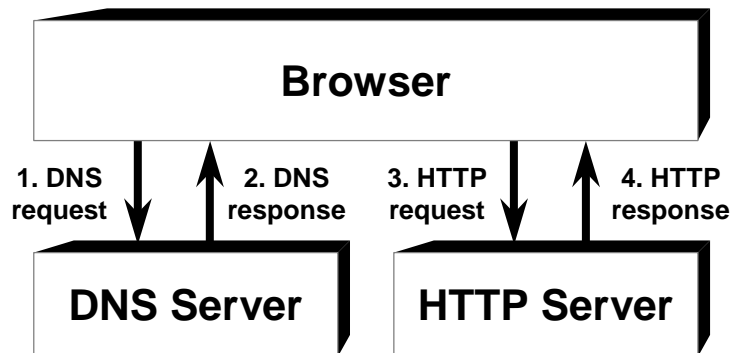
```
<p><a name="telefon">Telefonnummern:</a>
<p id="telefon">Telefonnummern:
```

- Identifizierung auf unterschiedlichen Ebenen
  - Buchstaben, Elemente, markierte Elemente
- Fragmente werden vom Client identifiziert
  - Uebertragung der kompletten Ressource
  - Lokalisierung des Fragments beim Client

## Internet Protocol Suite



## Verwendung von HTTP



WWW (SS2001) - HTTP

13

## Schema einer HTTP Interaktion

- *Verbindungsaufbau*
  - Aufbau TCP-Verbindung vom Client zum Server
  - normalerweise Port 80 des Servers
- *Request* vom Client zum Server
  - Auswahl einer Methode
  - zusätzliche Parameter zur Methode
- *Response* vom Server zum Client
  - Resultat in Form eines Statuscodes
  - zusätzliche Parameter zum Resultat
- *Verbindungsabbau*
  - normalerweise Abbau der Verbindung
  - neue Versionen (ab HTTP/1.1) können die Verbindung länger offen lassen

WWW (SS2001) - HTTP

14

## Beispiel für einen HTTP Request

- TCP-IP Verbindung zu www.ethz.ch
- Erstellen der Verbindung "von Hand"

```
> telnet www.ethz.ch 80
GET / HTTP/1.1
Host: www.ethz.ch
Accept-Language: en-US en de
```

## Beispiel für einen HTTP Response

- Akzeptieren der Verbindung vom Client
- Interpretation des Requests

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/4.0
Content-Location: http://www.ethz.ch/home_en.html
Date: Mon, 02 Nov 1998 09:24:53 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Fri, 30 Oct 1998 07:55:15 GMT
ETag: "74335a1da3be1:1b1a5"
Content-Length: 4229

<!DOCTYPE...
```

## HTTP Messages

- gehorchen einem allgemeinen Format
- sind entweder Requests oder Responses

```
message = start-line
          *message-header
          [ message body ]
```

- *start-line* bestimmt die Bedeutung
- *message-headers* enthält zusätzliche Information zur Message
- *message-body* enthält Daten

## HTTP Request

- Request vom Client zum Server

```
Methode Request-URI HTTP/major.minor
[Header]

[Entity]
```

- die Leerzeile muss vorhanden sein
- *minor*-Versionen sind kompatibel, *major*-Versionen sind inkompatibel
- Request meist ohne [Entity]
- [Header] enthält unterschiedliche Header
  - General Header, Request Header, Entity Header

## HTTP Response

- Response vom Server zum Client

```
HTTP/major.minor Statuscode Text  
[Header]
```

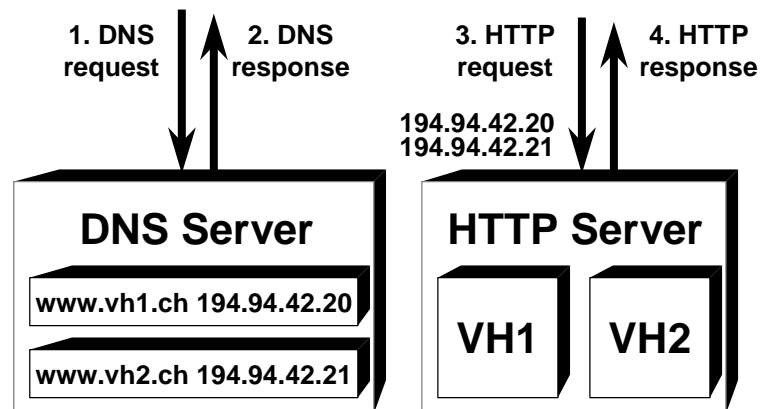
```
[Entity]
```

- die Leerzeile muss vorhanden sein
- Server muss sich an Client anpassen
- Response meist mit [Entity]
- [Header] enthält unterschiedliche Header
  - General Header, Response Header, Entity Header

## HTTP Header Häufigkeiten

	<i>Header Field Name</i>	<i>Frequency</i>	<i>Percent</i>
1.	Content-Type	25960	99.89%
2.	Server	25951	99.86%
3.	Date	25679	98.81%
4.	Last-Modified	22353	86.01%
5.	Content-Length	20945	80.95%
6.	Accept-Range	9666	37.19%
7.	Connection	4780	18.39%
8.	ETag	4033	15.52%

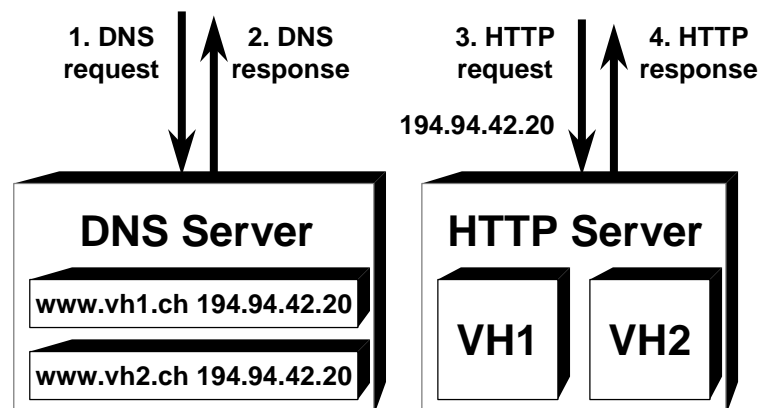
## IP-basierte virtuelle Hosts



WWW (SS2001) - HTTP

21

## Nicht-IP-basierte virtuelle Hosts



WWW (SS2001) - HTTP

22

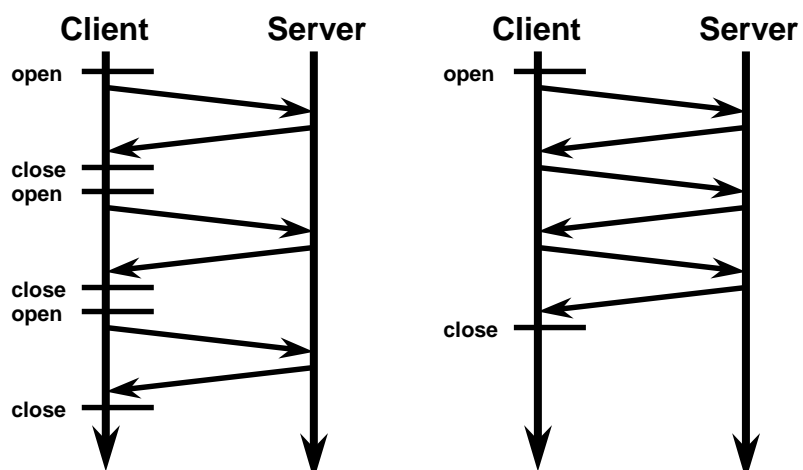
## HTTP/1.0 Performance-Probleme

- eine TCP Verbindung pro URL
  - viele teure Verbindungserstellungen
  - meist kurze Dokumente werden übertragen
  - viele alte Verbindungskontrollblöcke
- keine *Flow Control* für TCP Open und Close
  - bisher: Öffnen vieler simultaner Verbindungen
  - möglicherweise *Congestion* auf Links
- keine virtuellen Server möglich
  - individuelle IP-Adresse pro Server erzeugt u.U. Routing-Probleme
- Caching ist nur sehr einfach und fehlerhaft implementiert

WWW (SS2001) - HTTP

23

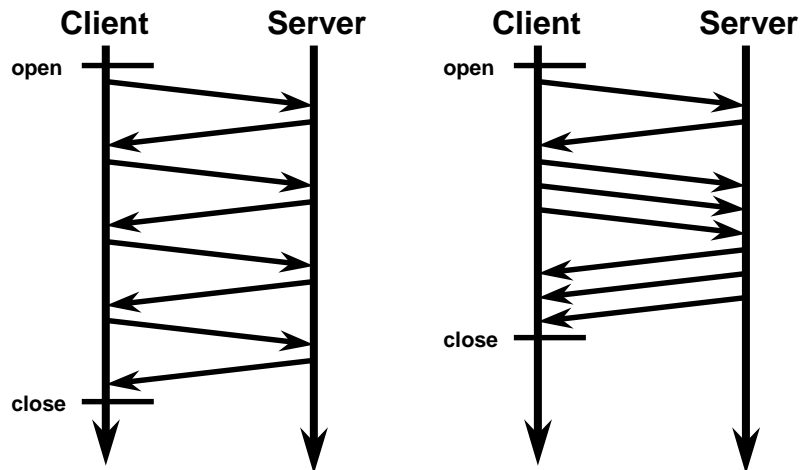
## Persistent Connections



WWW (SS2001) - HTTP

24

## Pipelining



WWW (SS2001) - HTTP

25

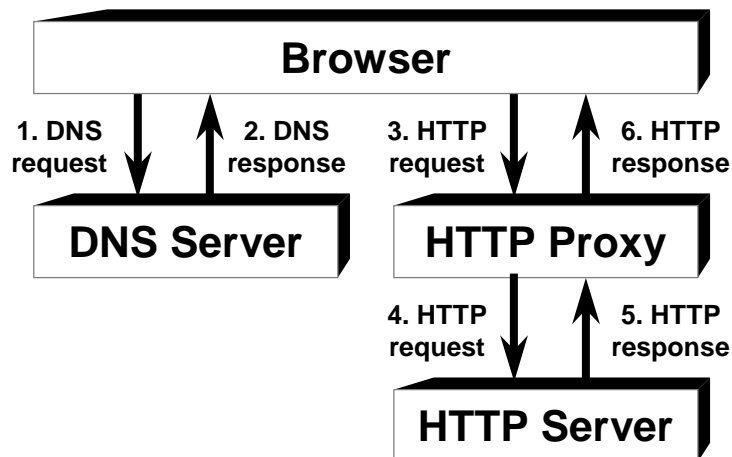
## HTTP/1.1 Methoden

- OPTIONS fragt Fähigkeiten des Servers ab
- GET holt eine Ressource vom Server
- HEAD liefert nur die Metainformationen zu einer Ressource
- PUT speichert eine Ressource
- POST liefert eine Menge von Daten an eine Ressource (CGI-Script)
- DELETE löscht eine Ressource
- TRACE ermöglicht das Verfolgen der Bearbeitung von Requests

WWW (SS2001) - HTTP

26

## HTTP Proxy



WWW (SS2001) - HTTP

27

## Aufgaben eines Proxy Servers

- *Caching* für eine Menge von Benutzern
  - typischerweise Platzierung vor einem *Bottleneck*
  - ISPs können gut profitieren von Caches
  - Nachteil: Cache muss explizit konfiguriert werden
- Platzierung in einem *Firewall* (Teil des Firewall)
  - Entscheidungen auf Applikationslevel möglich
  - Weiterleiten/Akzeptieren unbedenklicher Requests
  - ermöglichen lückenlose Kontrolle der Aktivitäten
  - aus Datenschutzsicht bedenklich
- Konfiguration eines Proxy sehr einfach
- aber: Cache-Konfiguration ständig überprüfen

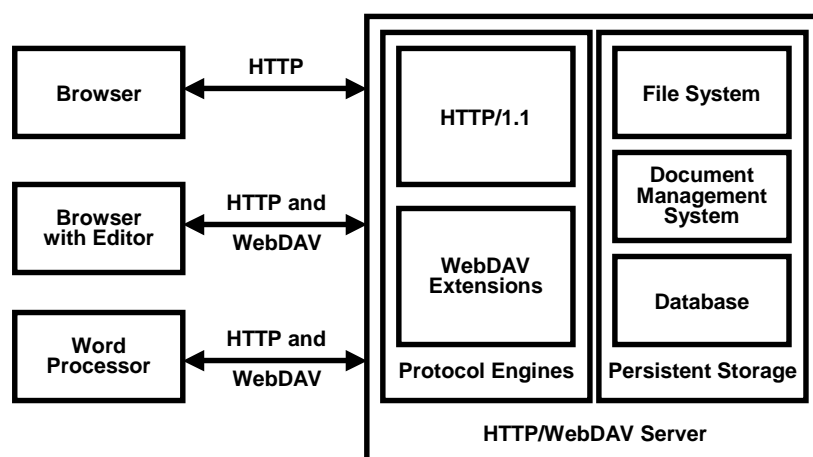
WWW (SS2001) - HTTP

28

## WebDAV

- HTTP kann zum Speichern verwendet werden
  - PUT Methode dient zum Speichern einer Ressource
  - auf nahezu allen Servern ausgeschaltet
  - Problem des verteilten Zugriffs auf Ressourcen
- Distributed Authoring and Versioning (WebDAV)
  - löst das Problem des verteilten Zugriffs
  - eine Erweiterung von HTTP
  - implementiert von Microsoft
  - soll proprietäre Lösungen ersetzen (Interoperabilität)
- spezifiziert in RFC 2518 (2/99)

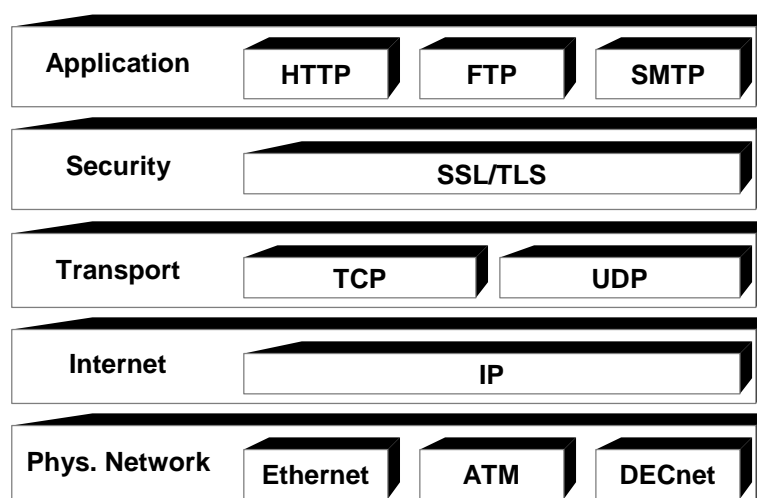
## WebDAV Szenario



## Secure Sockets Layer (SSL)

- z.Z. weitestverbreitete Sicherheitstechnologie
- Web-Server akzeptiert TCP/IP und TCP/IP über SSL Verbindungen
  - Kennzeichnung sicherer Dokumente mit dem `https:` URL Prefix
  - Standard-Verbindungen auf Port 80
  - SSL-Verbindungen auf Port 443
- aktuelle Version ist SSL3 (Verbesserungen gegenüber SSL2, mehr Algorithmen)
- Internet Standardisierung als *Transport Layer Security (TLS)*

## Internet Protocol Suite



## Zusammenfassung

---

- Ressourcenidentifikation im WWW
  - URI als allgemeiner Mechanismus
  - URL als weitverbreiteter Mechanismus
  - URL Schemes für verschiedene Dienste
- Ressourcenübertragung im WWW
  - HTTP als Client-Server-Protokoll
  - Message Typen von HTTP
  - Performance von HTTP
  - Caches und Proxies
- WebDAV für verteilten Schreibzugriff
- HTTPS als sichere Anwendung von HTTP