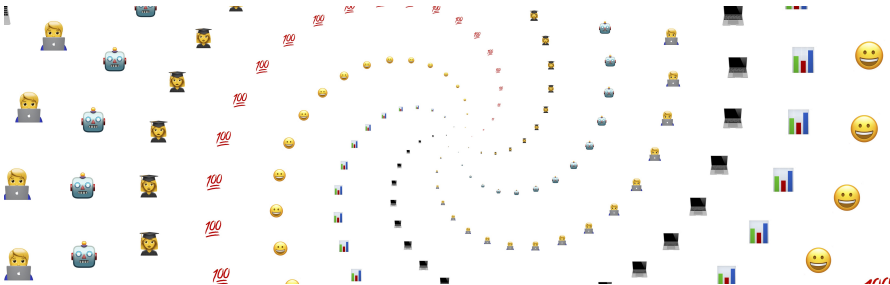
Netzwerke

Anwendungsschicht (Skript, Abschnitte 1.6 bis 1.8)

Cyril Blum

Fachschaft Informatik
Kantonsschule im Lee



Netzwerk-Spiel: **Anwendungsschicht**

Endgeräte verfassen Nachrichten:

Nachricht (max. 10 Zeichen): `Hi Bob, wi`

Nachricht (max. 10 Zeichen): `e geht's?`



Client vs Server vs Host

Anwendungsschicht

Definitionen

- **Client**



Client vs Server vs Host

Anwendungsschicht

Definitionen

- **Client**
  - „Kunde“ (von en. *client*)



Client vs Server vs Host

Definitionen

- **Client**
  - „Kunde“ (von en. *client*)
  - Computer(-Programm), das etwas will



Client vs Server vs Host

Definitionen

- **Client**
  - „Kunde“ (von en. *client*)
  - Computer(-Programm), das etwas will
- **Server**



Client vs Server vs Host

Definitionen

- **Client**
  - „Kunde“ (von en. *client*)
  - Computer(-Programm), das etwas will
- **Server**
  - „Diener“ (von en. *to serve* = bedienen, $\neq$ „surf“)



Client vs Server vs Host

Definitionen

- **Client**
  - „Kunde“ (von en. *client*)
  - Computer(-Programm), das etwas will
- **Server**
  - „Diener“ (von en. *to serve* = bedienen, $\neq$ „surf“)
  - Computer(-Programm), das etwas hat, das andere Computer(-Programme) brauchen



Client vs Server vs Host

Definitionen

- **Client**
  - „Kunde“ (von en. *client*)
  - Computer(-Programm), das etwas will
- **Server**
  - „Diener“ (von en. *to serve* = bedienen, $\neq$ „surf“)
  - Computer(-Programm), das etwas hat, das andere Computer(-Programme) brauchen
  - Kann Verbindungsanfragen von anderen Computer(-Programmen) akzeptieren / ablehnen



Client vs Server vs Host

Definitionen

- **Client**
  - „Kunde“ (von en. *client*)
  - Computer(-Programm), das etwas will
- **Server**
  - „Diener“ (von en. *to serve* = bedienen, $\neq$ „surf“)
  - Computer(-Programm), das etwas hat, das andere Computer(-Programme) brauchen
  - Kann Verbindungsanfragen von anderen Computer(-Programmen) akzeptieren / ablehnen
- **Host**



Client vs Server vs Host

Definitionen

- **Client**
  - „Kunde“ (von en. *client*)
  - Computer(-Programm), das etwas will
- **Server**
  - „Diener“ (von en. *to serve* = bedienen, ≠ „surf“)
  - Computer(-Programm), das etwas hat, das andere Computer(-Programme) brauchen
  - Kann Verbindungsanfragen von anderen Computer(-Programmen) akzeptieren / ablehnen
- **Host**
  - = Computer in einem Netzwerk, allgemein für „server“ oder „client“



Client vs Server vs Host

Definitionen

- **Client**
  - „Kunde“ (von en. *client*)
  - Computer(-Programm), das etwas will
- **Server**
  - „Diener“ (von en. *to serve* = bedienen, ≠ „surf“)
  - Computer(-Programm), das etwas hat, das andere Computer(-Programme) brauchen
  - Kann Verbindungsanfragen von anderen Computer(-Programmen) akzeptieren / ablehnen
- **Host**
  - = Computer in einem Netzwerk, allgemein für „server“ oder „client“
  - *Host name* = Name eines Computers



Browser

Anwendungsschicht

- Browser = Programm, kann eine Datei (meist `.html`)...
  - ...über das Internet *anfordern*
  - ...nach dem Empfang *darstellen*
- Das Senden / Empfangen von Dateien übernimmt das Betriebssystem (z.B. Windows, MacOS)
- Beispiele:
  - Chrome
  - Firefox
  - Safari
  - Edge
  - etc.



Domain Name System (DNS)

- Wenn Sie auf eine Webseite wie `https://sbb.ch` gehen, woher weiss der Client, von welchem Server er die Informationen anfragen soll?
- Domain Name System (DNS): Domain-Name $\rightleftarrows$ IP-Adresse!



FreeFlower

DNS

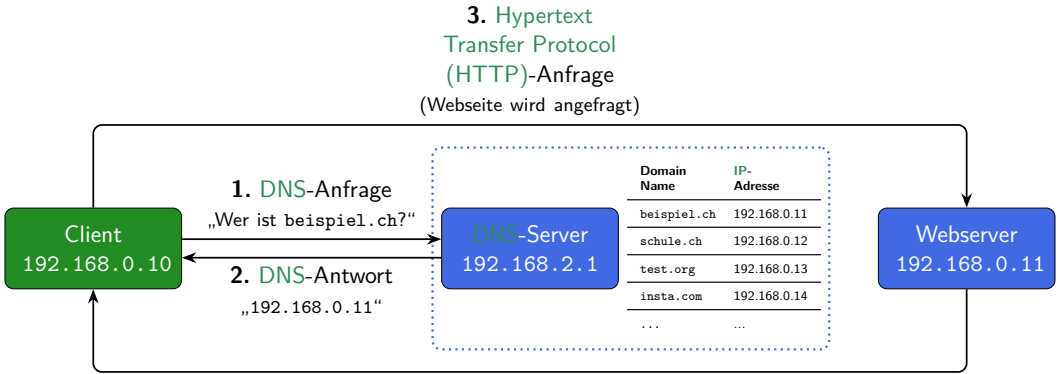


Abbildung: Schematischer Ablauf einer DNS-Anfrage

FreeFlower

URL (Uniform Resource Locator)

URL = Web-Adresse

$$\underbrace{\text{http://}}_{\text{Protokoll}}\ \underbrace{\text{www.}}_{\text{Subdomain}}\ \underbrace{\text{beispiel.}}_{\text{Domain}}\ \underbrace{\text{ch}}_{\text{TLD}}\ \underbrace{\text{/dokumente/}}_{\text{Ordner}}\ \underbrace{\text{reglemente.html}}_{\text{Dateiname}}$$

- Erster Teil bezeichnet einen *Server*

URL (Uniform Resource Locator)

URL = Web-Adresse

$$\underbrace{\text{http://}}_{\text{Protokoll}}\ \underbrace{\text{www.}}_{\text{Subdomain}}\ \underbrace{\text{beispiel.}}_{\text{Domain}}\ \underbrace{\text{ch}}_{\text{TLD}}\ \underbrace{\text{/dokumente/}}_{\text{Ordner}}\ \underbrace{\text{reglemente.html}}_{\text{Dateiname}}$$

- Erster Teil bezeichnet einen *Server*
  - z.B. `www.beispiel.ch` → 58.55.110.9

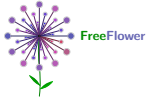


URL (Uniform Resource Locator)

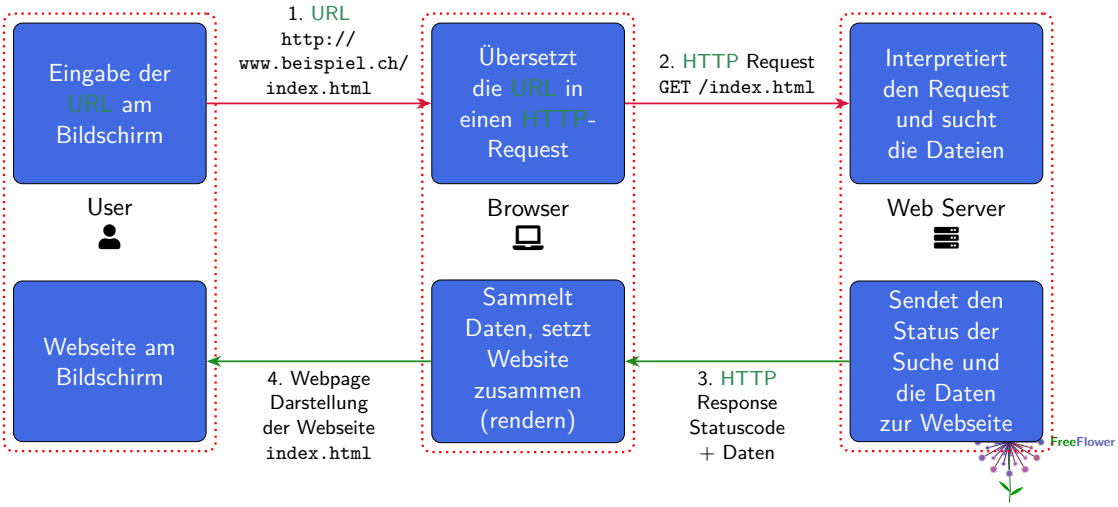
URL = Web-Adresse

$$\underbrace{\text{http://}}_{\text{Protokoll}}\ \underbrace{\text{www.}}_{\text{Subdomain}}\ \underbrace{\text{beispiel.}}_{\text{Domain}}\ \underbrace{\text{ch}}_{\text{TLD}}\ \underbrace{\text{/dokumente/}}_{\text{Ordner}}\ \underbrace{\text{reglemente.html}}_{\text{Dateiname}}$$

- Erster Teil bezeichnet einen *Server*
  - z.B. `www.beispiel.ch` → 58.55.110.9
- Alles nach der TLD ist ein normaler Ordnerpfad auf dem Computer



HTTP(S)-Protokoll



Beispiel: HTML (*Hypertext Markup Language*)

- Demo (*Inspect Element* oder *View Page Source*)



Beispiel: HTML (*Hypertext Markup Language*)

- Demo (*Inspect Element* oder *View Page Source*)
- Speichert die Struktur einer Webseite



Beispiel: HTML (*Hypertext Markup Language*)

- Demo (*Inspect Element* oder *View Page Source*)
- Speichert die Struktur einer Webseite
- Kann weitere Dateien enthalten, beispielsweise:



Beispiel: HTML (*Hypertext Markup Language*)

- Demo (*Inspect Element* oder *View Page Source*)
- Speichert die Struktur einer Webseite
- Kann weitere Dateien enthalten, beispielsweise:
  - CSS (Cascading Style Sheet), für die Darstellung (Farbe, Grösse etc.)



Beispiel: HTML (*Hypertext Markup Language*)

- Demo (*Inspect Element* oder *View Page Source*)
- Speichert die Struktur einer Webseite
- Kann weitere Dateien enthalten, beispielsweise:
  - CSS (Cascading Style Sheet), für die Darstellung (Farbe, Grösse etc.)
  - JavaScript, jQuery etc. für die Interaktivität (Ein- und Ausblenden von Objekten etc.)



Beispiel: HTML (*Hypertext Markup Language*)

- Demo (*Inspect Element* oder *View Page Source*)
- Speichert die Struktur einer Webseite
- Kann weitere Dateien enthalten, beispielsweise:
  - CSS (Cascading Style Sheet), für die Darstellung (Farbe, Grösse etc.)
  - JavaScript, jQuery etc. für die Interaktivität (Ein- und Ausblenden von Objekten etc.)
- Wie man eine Webseite baut ist hier nicht das Hauptthema, wird aber in der Interessenwoche vermittelt



Webserver *ohne* DNS

- Netzlabor (Web-Server): 1.35
- Challenge: 1.36 (HTML besser verstehen)
- Challenge: Netzlabor (Web-Server via Modem): 1.37

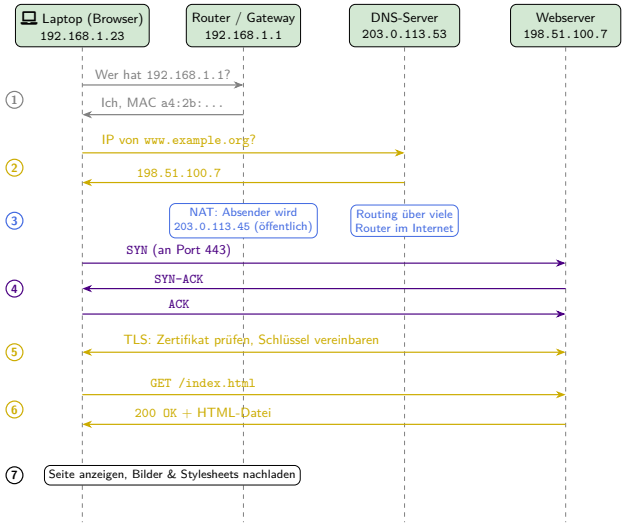
DNS

- 1.38 - 1.40

FreeFlower

Ablauf eines Webaufrufs

Alle Schichten arbeiten zusammen

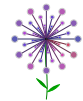


FreeFlower

Ablauf eines Webaufrufs

Schritt für Schritt

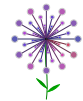
1. **Netzzugang**: MAC-Adresse des Routers mit ARP erfragen



Ablauf eines Webaufrufs

Schritt für Schritt

1. **Netzzugang**: MAC-Adresse des Routers mit ARP erfragen
2. **Anwendung**: IP-Adresse des Webservers mit DNS erfragen



Ablauf eines Webaufrufs

Schritt für Schritt

1. **Netzzugang**: MAC-Adresse des Routers mit ARP erfragen
2. **Anwendung**: IP-Adresse des Webservers mit DNS erfragen
3. **Vermittlung**: NAT im Heimrouter, Routing durchs Internet



FreeFlower

Ablauf eines Webaufrufs

Schritt für Schritt

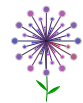
1. **Netzzugang**: MAC-Adresse des Routers mit ARP erfragen
2. **Anwendung**: IP-Adresse des Webservers mit DNS erfragen
3. **Vermittlung**: NAT im Heimrouter, Routing durchs Internet
4. **Transport**: TCP-Verbindung zu Port 443 (3-way handshake)



Ablauf eines Webaufrufs

Schritt für Schritt

1. **Netzzugang**: MAC-Adresse des Routers mit ARP erfragen
2. **Anwendung**: IP-Adresse des Webservers mit DNS erfragen
3. **Vermittlung**: NAT im Heimrouter, Routing durchs Internet
4. **Transport**: TCP-Verbindung zu Port 443 (3-way handshake)
5. **Anwendung**: TLS – Zertifikat prüfen, Schlüssel vereinbaren

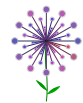


FreeFlower

Ablauf eines Webaufrufs

Schritt für Schritt

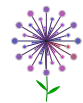
1. **Netzzugang**: MAC-Adresse des Routers mit ARP erfragen
2. **Anwendung**: IP-Adresse des Webservers mit DNS erfragen
3. **Vermittlung**: NAT im Heimrouter, Routing durchs Internet
4. **Transport**: TCP-Verbindung zu Port 443 (3-way handshake)
5. **Anwendung**: TLS – Zertifikat prüfen, Schlüssel vereinbaren
6. **Anwendung**: HTTP-Anfrage `GET`, Antwort 200 `OK` mit HTML



Ablauf eines Webaufrufs

Schritt für Schritt

1. **Netzzugang**: MAC-Adresse des Routers mit ARP erfragen
2. **Anwendung**: IP-Adresse des Webservers mit DNS erfragen
3. **Vermittlung**: NAT im Heimrouter, Routing durchs Internet
4. **Transport**: TCP-Verbindung zu Port 443 (3-way handshake)
5. **Anwendung**: TLS – Zertifikat prüfen, Schlüssel vereinbaren
6. **Anwendung**: HTTP-Anfrage `GET`, Antwort 200 `OK` mit HTML
7. Seite darstellen, Bilder & Stylesheets nachladen



Auftrag (Skript)

- 1.41 (Ablauf eines Webaufrufs)
- Challenge: 1.42 (Webaufruf in Wireshark)
- Challenge: 1.45 (Troubleshooting)
- Challenge: 1.46 (Vertiefung Schichtenmodell)

