# Funktionsweise des Webs

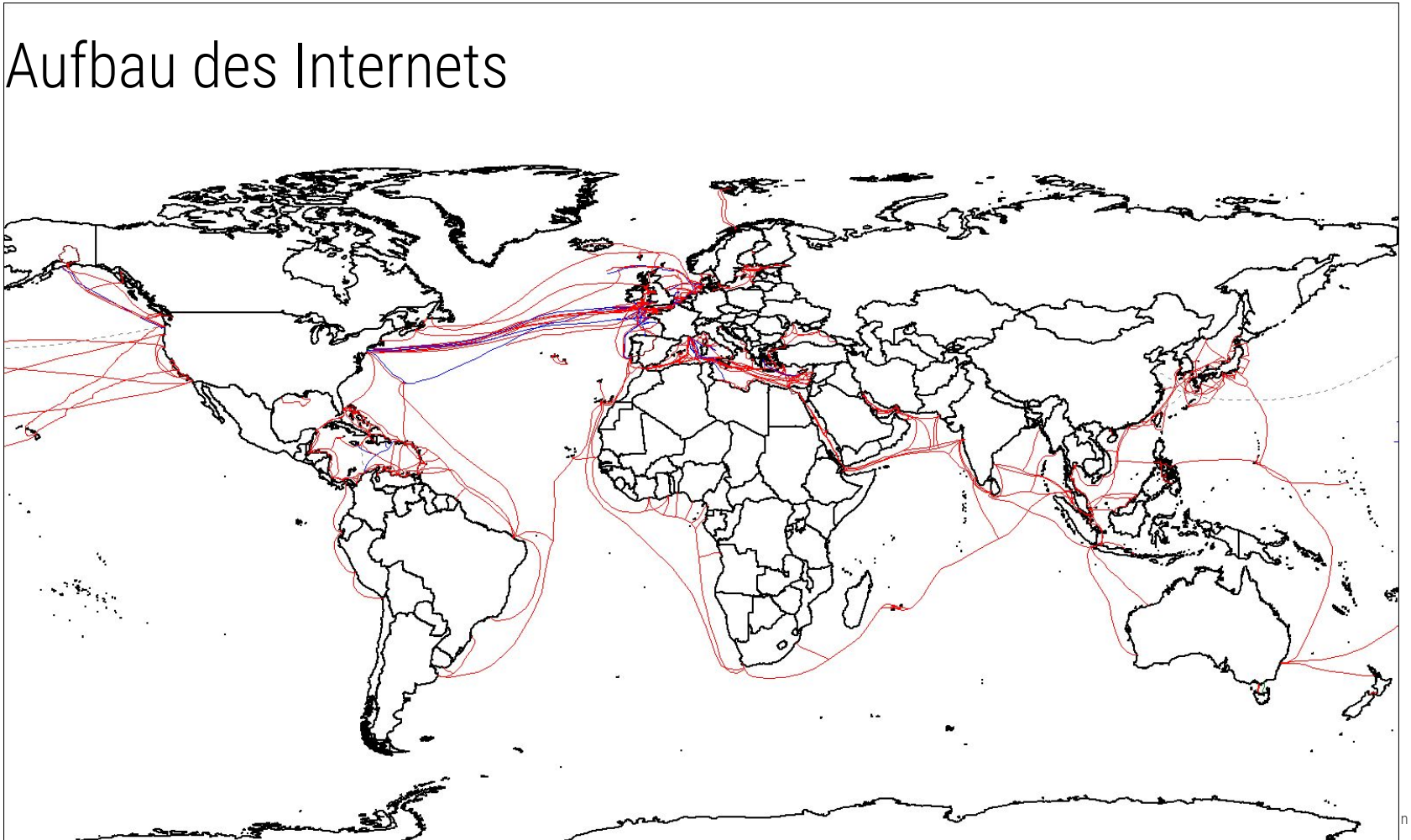
# Geschichte des Internet

- Ab 1967 entwickelt durch ARPA als ARPANET
  - Verband zuerst einzelne US-amerikanische Universitäten miteinander
- 1973 erste internationale Verbindungen
  - Norwegen und London
- 1978 erste Spezifikation von TCP/IP als getrennte Protokolle
  - **Gab inhaltliche Verantwortung an die Hosts im Netzwerk**
    - Massiver Schritt in Richtung Massentauglichkeit und Konnektivität durch Flexibilität
    - Macht das Internet zu **“Netzwerk aus Netzwerken”**
- **1983 IPv4** wird im ARPANET eingeführt
  - Aktuelle Protokoll-Basis des Internets
  - IPv6 übernimmt gerade im Parallelbetrieb
- 1995 Kommerzialisierung des Internets in den USA

# Aufbau des Internets

- Netzwerk aus Netzwerken
- Verbunden durch Router
  - Router stellen Verbindung zwischen einzelnen Netzwerken her
- Routing findet innerhalb einer (organisatorischen) Hierarchie statt
  - Der Netzwerkverbund ist aus mehrere Schichten aufgeteilt
  - Siehe z.B. interkontinentale Netzwerke
  - Dienen zur Verwaltung und kommerziellen Gebührenerhebung (Basis z.B. der Telekom)

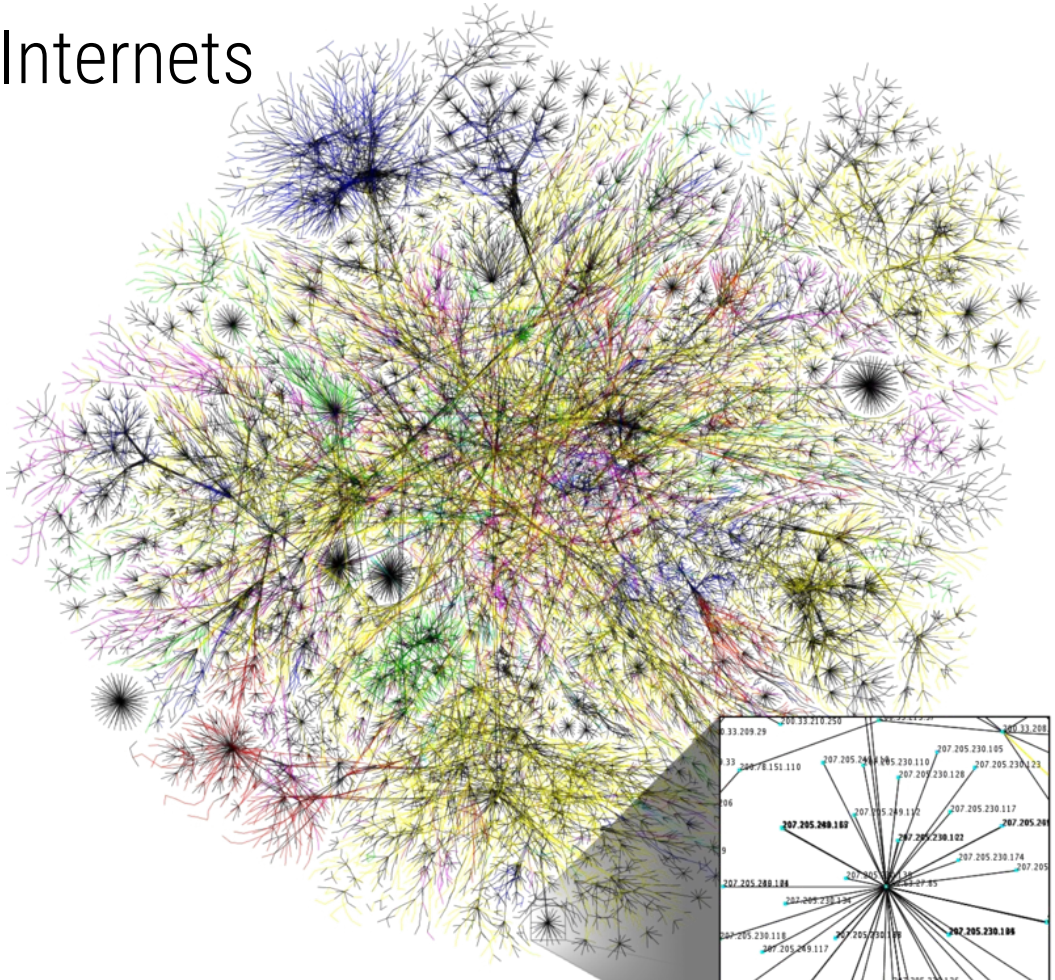
# Aufbau des Internets



# Aufbau des Internets - IP Protokoll

- Erlaubt Routing von Daten im Internet
- Hosts im Netzwerk werden durch die IP Adresse identifiziert
- 32 bit (128 bit für IPv6) lange Adresse
- IP Adresse ist aufgeteilt in einzelne Nummern
  - Beinhaltet Teile der Routing Hierarchie
  - Erlaubt Blocken von Verbindungen über Netzwerkgrenzen hinweg
- Mehr oder weniger einfach
  - 8.8.8.8 Public Google DNS
  - 2a02:2e0:3fe:1001:7777:772e:2:85 Webhosting www.heise.de

# Aufbau des Internets



# Geschichte des Webs (WWW)

- 1983 Entwicklung des Domain Name System (DNS)
  - **Domains statt IP Adressen**
- 1990 Tim Berners-Lee schlägt das “WorldWideWeb” als Medium zur Informationsweitergabe vor
  - Intern am CERN
  - Vorschlag beinhaltet bereits Ideen zu HTML, HTTP und Browsern
  - Querverlinkung von Dokumenten als Neuerung
- Dezember 1990 erste Website der Welt
- **1994 Spezifikation der URL**
- **1997 Spezifikation von HTTP**

**Das Web ist geboren**

# URL

- Uniform Resource Locator
- Ermöglicht eindeutige Identifikation von Ressource (z.B. einer Datei) im Internet
- Sonderform der URI
- Enthält:
  - Ort der Ressource
  - Möglichkeit diese abzurufen (über Protokoll und Webadresse)
- Für viele andere Netzwerkanwendungen genutzt
  - Z.B. FTP

# URL

- Aufgeteilt nach expliziter Syntax

Schema://[Userinfo@]Host[:Port]Pfad[?Query][#Fragment]

Beispiele:

https://www.th-rosenheim.de/die-hochschule/fakultaeten-institute/fakultaet-fuer-informatik/

https://duckduckgo.com/?q=web+entwicklung&t=canonical&ia=news

ftp://user:pwd@test.io/path/to/what/you/need

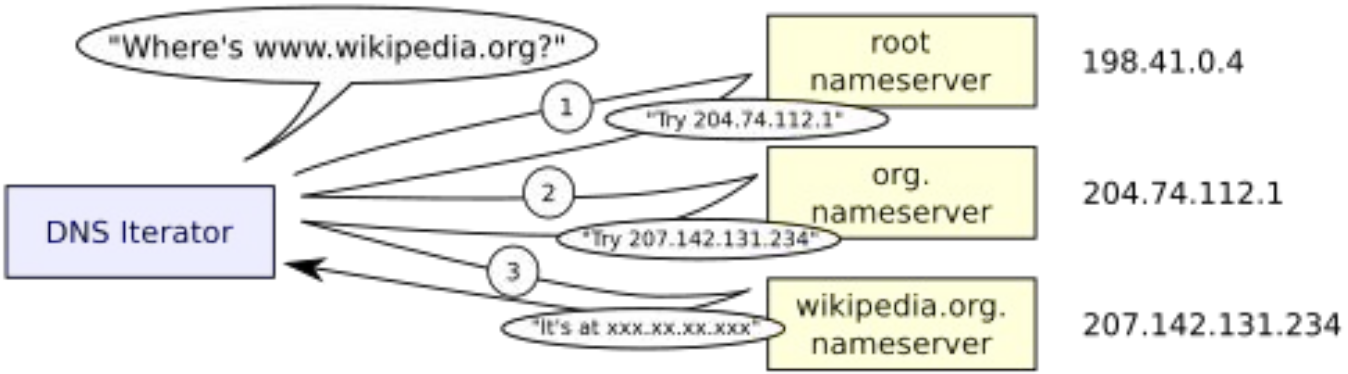
http://**例子**.**卷筒纸**

# URL - Bestandteile

- Schema: “Protokollfamilie” z.B. http, https, ftp
- Userinfo: Eventuelle Credentials (selten genutzt)
- Host: Die IP oder Domain
- Pfad: Ort der Ressource auf dem Server oder innerhalb der Anwendung (meist virtuell)
- Query
  - Key/Value Paare
  - Getrennt durch Delimiter
- Fragment
  - Im Web Kontext meistens eine Sprungmarke
  - Früher für Javascript Frameworks zur Navigation verwendet

# Domain Name System

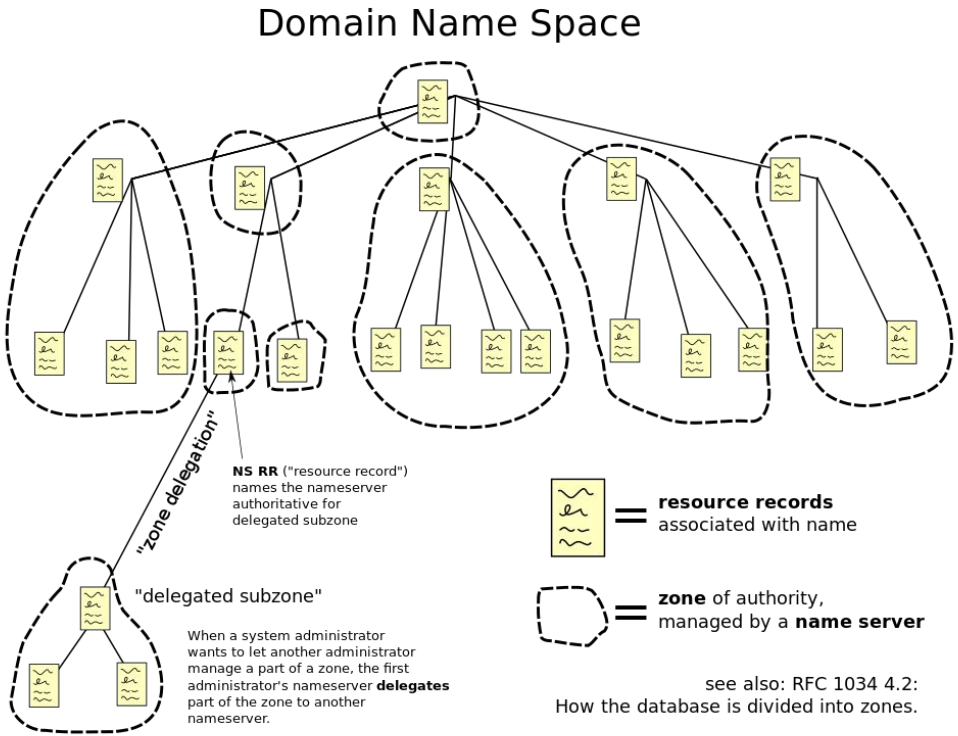
- Löst Domains in IP Adressen auf
- Hierarchisch aufgebaute
- Selbstsynchronisierend (Update kann dauern)



# Domain Name System - Aufbau

- Hierarchischer Aufbau
- Zentrales Management durch ICANN
  - Lokales Handling durch Registrare
  - Z.B. DENIC eG für .de
- Trennung (von rechts nach links) in:
  - Top Level Domain (TLD)
  - Second Level Domain (SLD)
  - X Level Domain (“Subdomains”)

www.th-rosenheim.de.



# HTTP Protokoll

- Application Layer Protokoll des TCP/IP Stacks
- Technische Basis des WWW
- Aufgeteilt in **Request** und **Response**
  - Gemacht für **synchrone** Kommunikation
  - Bedingt **Client/Server** Modell
- Ist als **stateless** konzipiert
- Kontinuierlich weiterentwickelt
- Verschlüsselbar (als HTTPS)

# HTTP Protokoll - Versionen

- Alle in großen Teilen rückwärtskompatibel

- 1997 HTTP/1.1
- 2015 HTTP/2
  - TCP Connection Multiplexing
  - Server Push
- ? HTTP/3
  - Verbesserung bei Head-of-Line BLocking

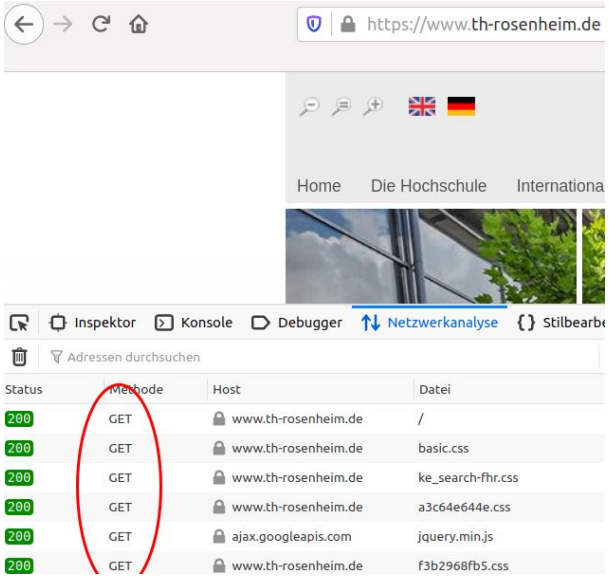
# HTTP Protokoll - Request

- Beispielrequest (aus RFC 7230)

**GET** /hello.txt HTTP/1.1
User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
Host: www.example.com
Accept-Language: en, mi

# HTTP Protokoll - Request Methods

- Standardisierte Request Semantik
  - WAS will ich mit meinem Request erreichen
- Teil der HTTP Status Zeile
- 8 Stück im Standard
- Prominenteste Methoden von Webanwendungen
  - GET
  - POST
- Andere Methoden interessant für Webservices
  - Sieh z.B. REST
- Maschinenlesbar



# HTTP Protokoll - Request Methods

## GET

- Genutzt für alle Abfragen
- Senden von Daten auch möglich
  - Enthält alle Daten als Teil des Query Strings in der URL
- Zum Teilen von Detailanfragen z.B. Google Maps
  - https://www.google.com/maps/place/Technische+Hochschule+Rosenheim/@47.8674599,12.1051211,17z/data=!3m1!4b1!4m5!3m4!1s0x47761bbaa49606ad:0x5a5d95627632ef5!8m2!3d47.8674563!4d12.1073098
- Häufigstes Methode im Web

# HTTP Protokoll - Request Methods

## POST

- Genutzt zum Senden von Information
  - Wie oder wozu Informationen verarbeiten werden entscheidet die Anwendung
  - Vor allem zum Senden von Formulardaten
  - Oft auch zum Anlegen neuer Datensätze (vergleiche REST CRUD)
- Enthält alle Daten innerhalb des Request Bodies
  - Nicht direkt teilbar

# HTTP Protokoll - Request

- Beispielrequest (aus RFC 7230)

GET /hello.txt HTTP/1.1

**User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3**

**Host: www.example.com**

**Accept-Language: en, mi**

Server response:

# HTTP Protokoll - Request/Response Header

- Essentielle Meta-Information
  - Sowohl Request als auch Response
- Steuern Verhalten von Client und Server
- Gebe Informationen zu Weg der Übertragung (z.B. Proxy Nutzung)
- Einige standardisierte Header
- Viele proprietäre
  - Jeder kann eigene Header setzen und verarbeiten (oft an Prefix “x-” erkennbar)
  - Obliegt Anwendungsentwicklern bzw. Browserherstellern

# HTTP Protokoll - Request Header

- Host: Der angefragte Host
- User-Agent: Identifikation des anfragenden Browsers
- Cookie: Kommunikation zu lokalem Authentizitäten-Speicher
- Accept: Welche Medientypen erwartet werden (z.B. Text)
- Accept-Language: Bevorzugte Sprachen des Nutzers
- Referer: Wo war der Nutzer zuvor?

# HTTP Protokoll - Response Header

- Content-Type: Medientype der Antwort (vergleiche “Accept” im Request)
- Server: Identification der Serversoftware
- Content-Encoding: Art der verwendeten Komprimierung
- Cache-Control: Caching Metainformation
- Strict-Transport-Security: HTTPS Sicherheitsoption

# HTTP Protokoll - Response

HTTP/1.1 **200** OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Content-Length: 12
Content-Type: text/plain

Hello World!

# HTTP - Status Codes

# HTTP - Status Codes

- Standardisierte Request Antwort des HTTP Protokolls (RFC 7231)
    - Kontrolliert durch Anwendungsentwickler
- Werden noch aktiv erweitert (zuletzt 2015)
- Aufgeteilt in 5 Kategorien mit festen Nummernkreisen
    - Informativ **1xx**
    - Erfolgsmeldungen **2xx**
    - Umleitungen **3xx**
    - Client Fehler **4xx**
    - Server Fehler **5xx**

Vorteil: **maschinenlesbar**

# HTTP - Status Codes

Einige wichtige Codes

200 OK - Request war erfolgreich

301 Moved Permanently - Die Ressource ist anderweitig (wie angezeigt) erreichbar

401 Unauthorized - Für die Ausführung des Requests ist eine Authentifizierung notwendig

403 Forbidden - Es fehlen nötige Rechte zur Ausführung des Requests

404 Not Found - Ressource existiert nicht (oder wird geheim gehalten)

500 Internal Server Error - Unbestimmter Fehler bei der Requestverarbeitung

**Bildquellen:**

https://en.wikipedia.org/wiki/Internet#/media/File:Internet_map_1024_-_transparent,_inverted.png
http://www.linux-onlineshop.de/store_files/1/images/product_images/popup_images/1ts_i_hacked.png
https://commons.wikimedia.org/wiki/File:Example_of_an_iterative_DNS_resolver.svg
https://commons.wikimedia.org/wiki/File:Domain_name_space.svg
https://http.cat/404