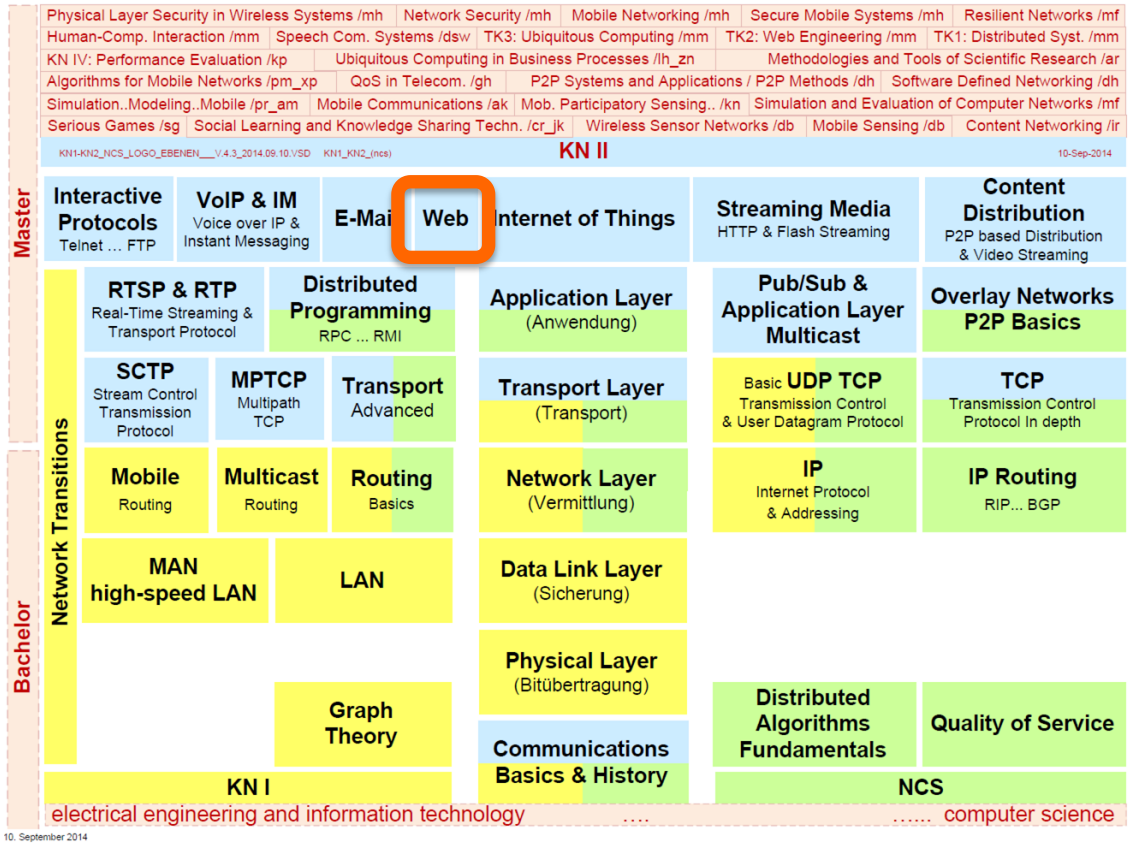
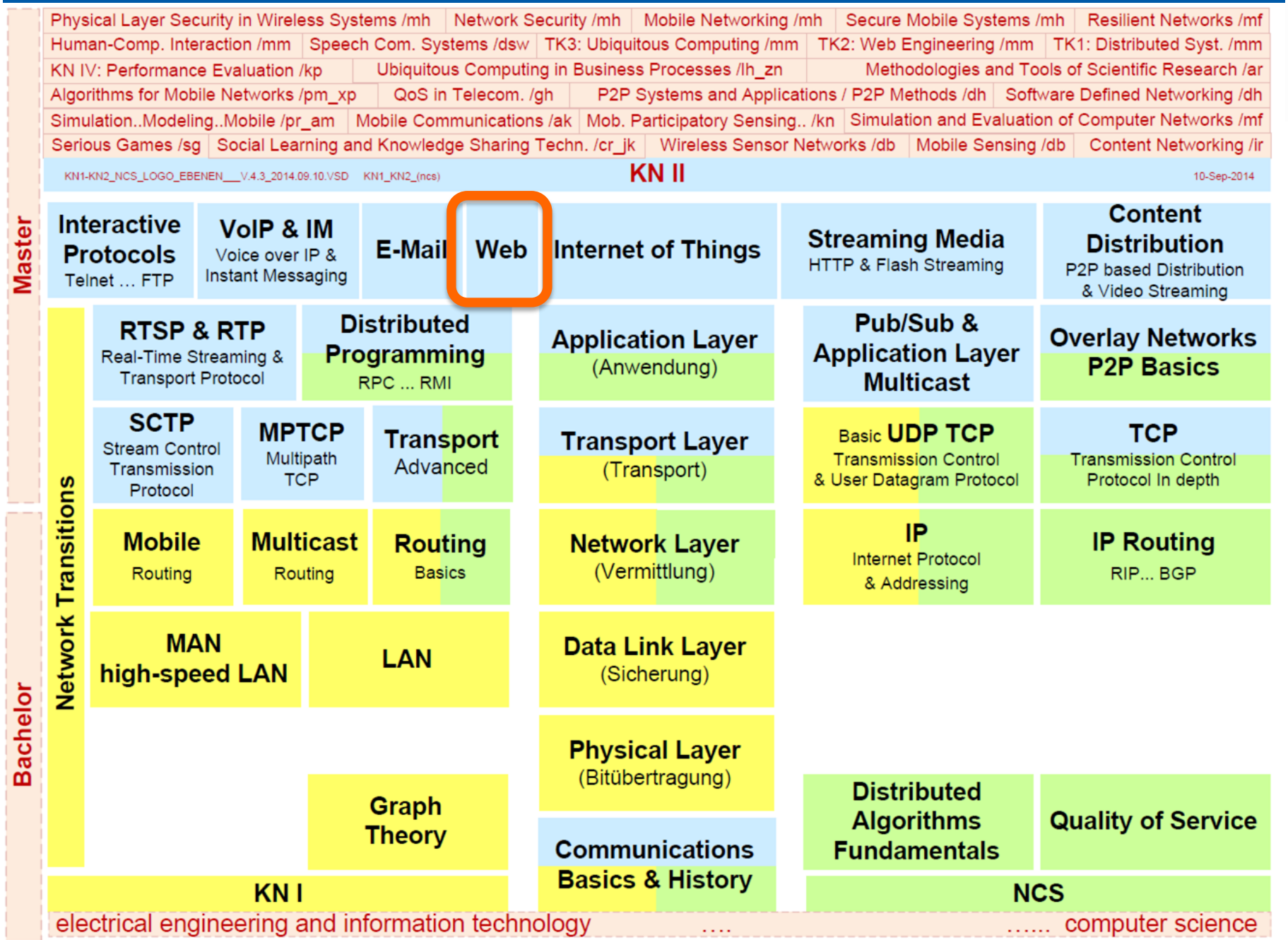
Communication Networks II

Web

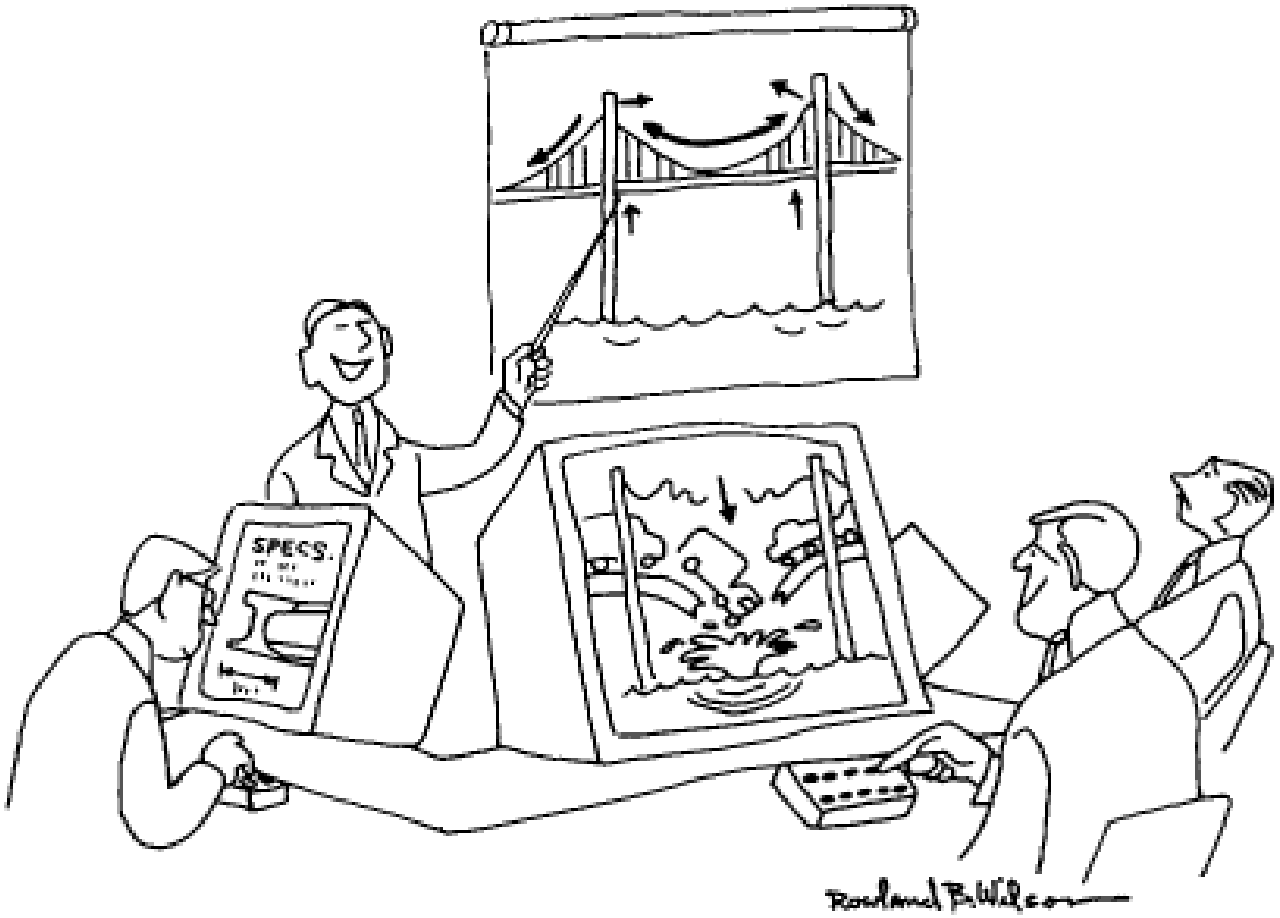


Prof. Dr.-Ing. **Ralf Steinmetz**
KOM - Multimedia Communications Lab

Overview



10. September 2014



Source: The Computer as a Communication Device, Science and Technology, 1968, reprint

The Computer as a Communication Device

In a few years, men will be able to communicate more effectively through a machine than face to face.

we believe that we are entering a technological age in which we will be able to interact with the richness of living information—not merely in the passive way that we have become accustomed to using books and libraries, but as active participants in an ongoing process, bringing something to it through our interaction with it, and not simply receiving something from it by our connection to it.

Such a medium is at hand—the programmed digital computer. Its presence can change the nature and value of communication even more profoundly than did the printing press and the picture tube, for, as we shall show, a well-programmed computer can provide direct access both to informational resources and to the *processes* for making use of the resources,

Web – Invented 1989

1989	Tim Berners-Lee (CERN, Geneva) publishes his first ideas
1993	approx. 50 web-servers
1993	NCSA distributes first version of Mosaic browser as shareware
1994	CERN and MIT found W3 Organization (see also http://www.w3.org)
…	
1995	HTML defined as HTML 2.0. in RFC 1866
1996	HTTP 1.0 defined in RFC 1945
1996	HTML 3.2 consensus for 1996
1998	HTML 4.0 and very few variations
1999	HTTP 1.1 defined in RFC 2616
…	
…	XML, CSS, … RSS feeds, …AJAX, REST, Mashups…
Today	HTML 5.x … web server everywhere
…	
Future	Semantic web to add “knowledge” to nodes (as metadata)

Source: The Computer as a Communication Device, Science and Technology, 1968, reprint

1.2 Browser Statistics

Quiz on worldwide usage

Out of 100%

Firefox ?

Chrome ?

Opera ?

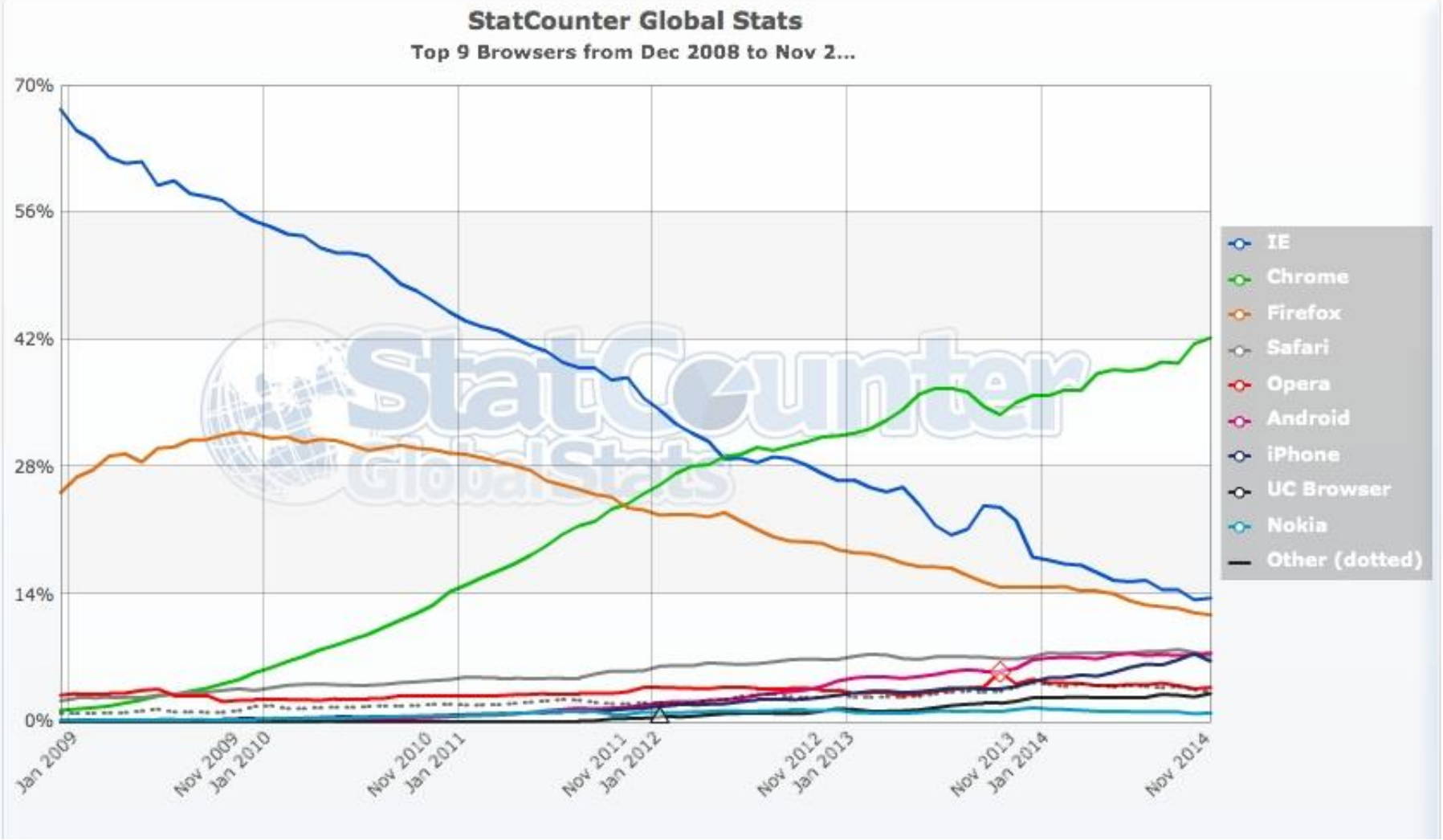
Internet Explorer ?

Safari ?

Further ?

http://gs.statcounter.com

Browser Statistics



1.3 Address - Uniform Resource Locators (URL)

TECHNISCHE UNIVERSITÄT DARMSTADT

URL is the „address“ of a page

Format: <SCHEME>:<SCHEME-SPECIFIC-PART>

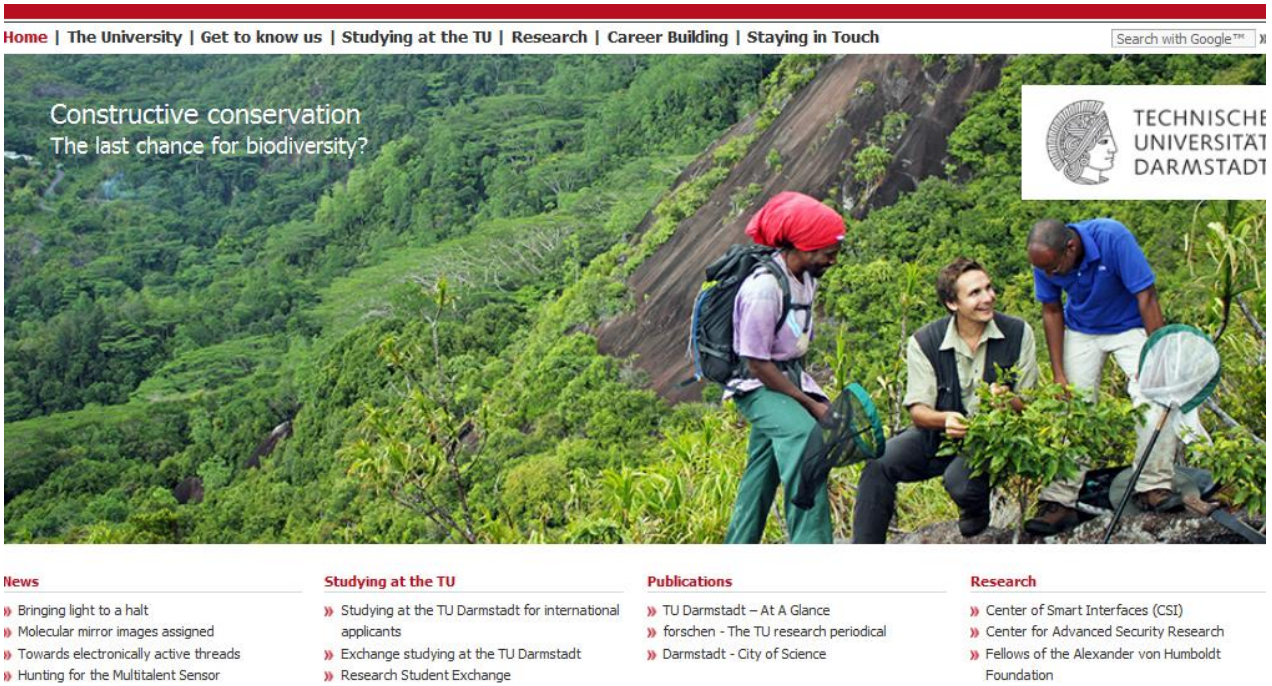
- http://<host>:<port>/<path>?<query_string>
- ftp://<user>:<password>@<host>:<cwd_1>/../<cwd_N>/<name>;type=<typecode>
- mailto:<rfc822-addr-spec>
- telnet://<user>:<password>@<host>:<port>
- file://<host>/path

	Used for	Example
Examples	Hypertext	http://www.kom.tu-darmstadt.de/
	FTP	ftp://ftp.ibr.cs.tu-bs.de/README
	Local file	file://home/rsteinmetz/.signature
	Sending email	mailto:kn2-lecture@kom.tu-darmstadt.de
	Remote login	telnet://www.w3.org:80

TECHNISCHE UNIVERSITÄT DARMSTADT

Web page consists of objects

- Object can be
  - HTML file
  - JPEG image
  - Java applet
  - Audio file,…



Web page consists of base HTML file

- Includes several referenced objects
- Each object is addressable by a URL (Uniform Resource Locator)
  - http://www.kom.tu-darmstadt.de/.....

Web pages are accessed using hypertext transfer protocol (HTTP)

TECHNISCHE UNIVERSITÄT DARMSTADT

Hypertext Markup Language (HTML)

- Based on standard generalized markup language (SGML)
  - Standardized by ISO
- Defines markup tags for structuring text
- Browser interprets tags and generates page layout

Example HTML tags

- <HEAD>...</HEAD> page header
- <BODY>…</BODY> page body
- <H1>…</H1> headline
- <P>…</P> paragraph
- <IMG SRC="..."> inserted image
- <A HREF="...">...</A> link to another document

Example file

```
<html>
    <body>
        <h1>Hello World!</h1>
    </body>
</html>
```

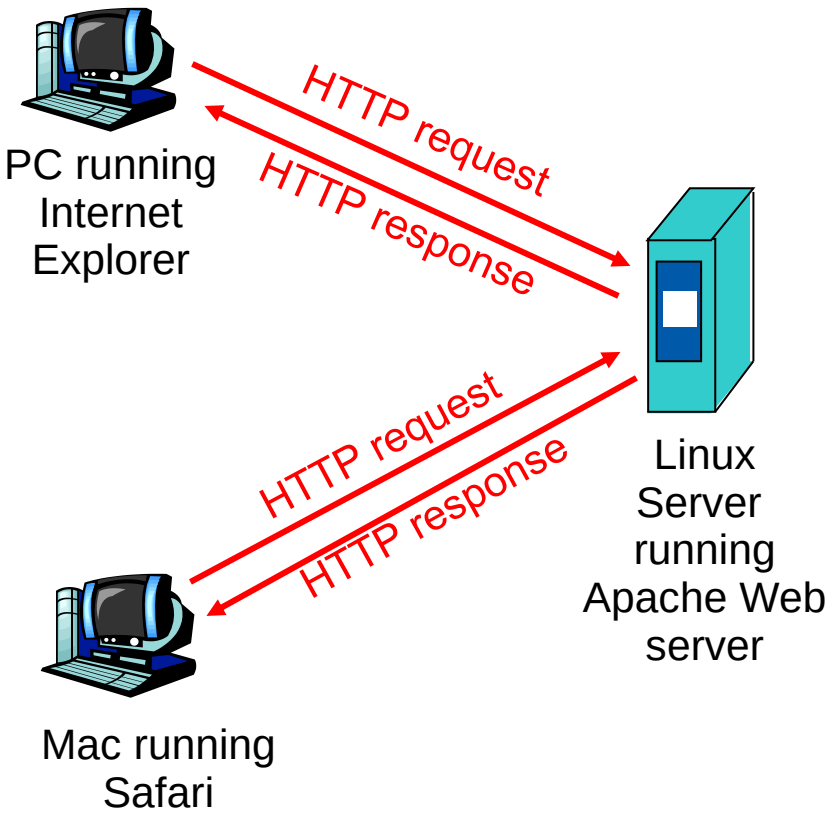
Defining hyperlinks

- Format: <A HREF="some URL"> hyperlinked text </A>
- Example code:
  Click <A HREF="http://www.tu-darmstadt.de"> here </A> to go to TU Darmstadt.
- Example layout:
  Click here to go to TU Darmstadt.

TECHNISCHE UNIVERSITÄT DARMSTADT

Hypertext transfer protocol (HTTP)

- Web's application layer protocol
- Client/server model
  - Client: browser that requests, receives, "displays" Web objects
  - Server: Web server sends objects
    - In response to requests



HTTP uses TCP

- Client initiates TCP connection
  - To serverport 80
- Server accepts TCP connection
- HTTP messages (application-layer protocol messages) exchanged
  - Between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is a stateless protocol

- Server maintains no information about past client requests
- Side note:
  protocols that maintain state are more complex
  - Past history (state) must be maintained
  - If server/client crashes
    - Their views of state may be inconsistent
    - State must be reconciled
  - Compare e.g. protocols in database world

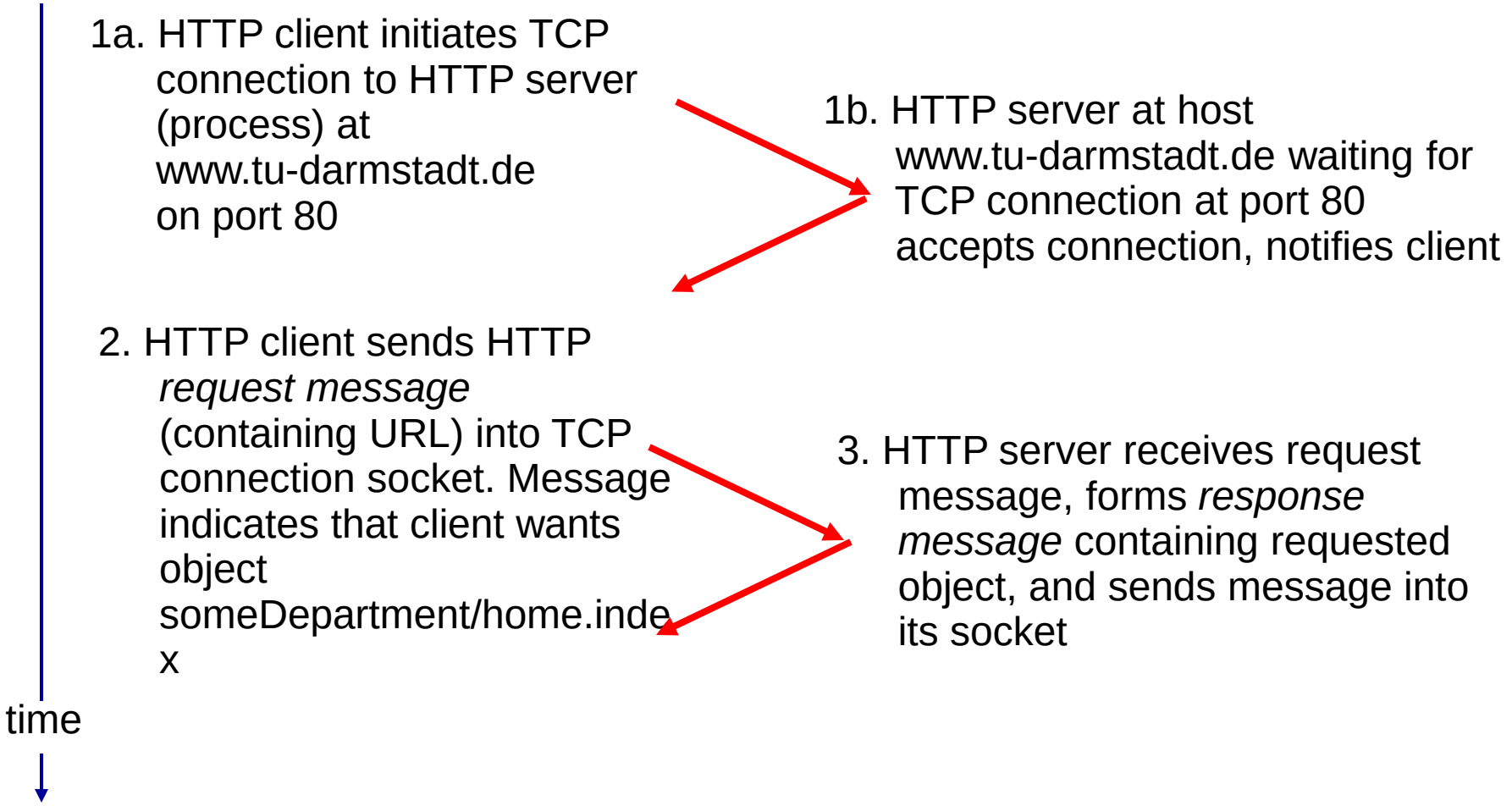
HTTP connections can be

- Non-persistent (HTTP 1.0)
  - At most one object sent over one TCP connection
- Persistent (HTTP 1.1)
  - Multiple objects can be sent over one TCP connection

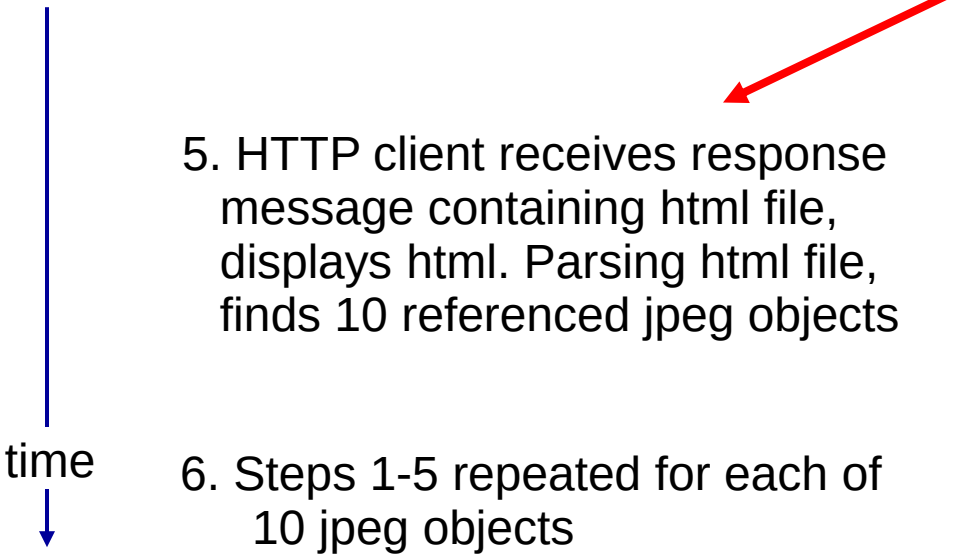
TECHNISCHE UNIVERSITÄT DARMSTADT

Suppose user enters URL http://www.tu-darmstadt.de

- Suppose page contains text and references to 10 jpeg images



Non-persistent HTTP



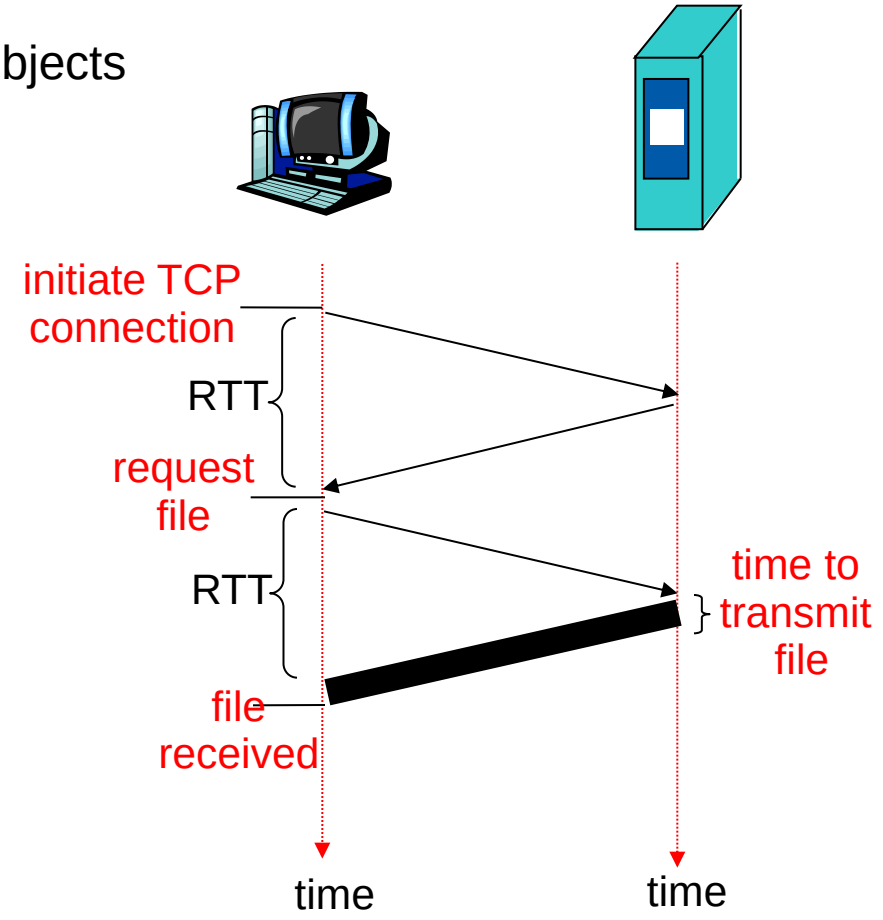
Non-persistent HTTP

A closer look at the response time

- For loading a web page with multiple objects
- Based on round trip time (RTT)
  - Between client and server

Response time consists of

- One RTT to initiate TCP connection
- One RTT for HTTP request
  - And first few bytes of HTTP response to return
- File transmission time
- Total = 2RTT + transmission time



Motivation: non-persistent HTTP issues

- Requires 2 additional RTTs per object
- Operating system overhead for each TCP connection
- Amplified by browsers opening parallel TCP connections
  - To fetch referenced objects

Persistent HTTP

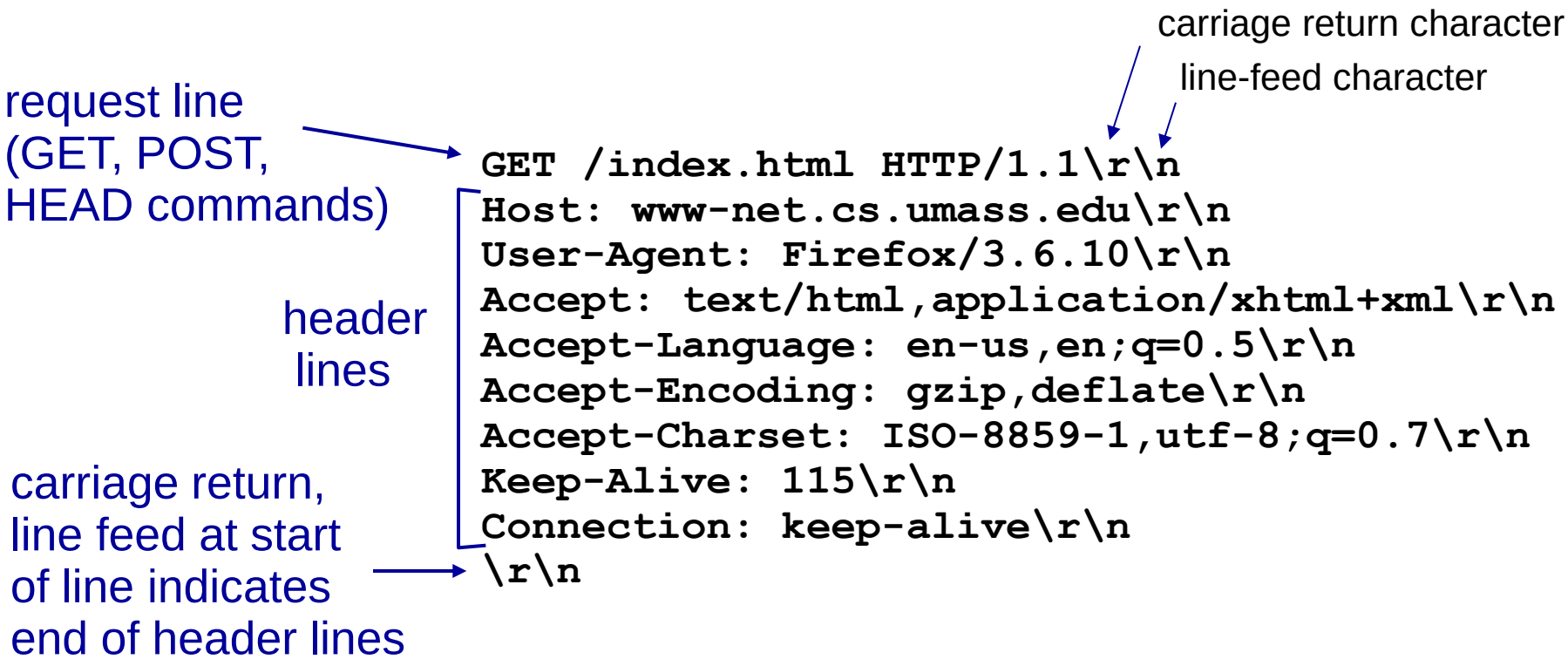
- Server leaves connection open
  - After sending response
- Subsequent HTTP messages sent over open connection
- Client sends requests as soon as it encounters a referenced object
- Overhead reduced to ~one RTT for all referenced objects

TECHNISCHE UNIVERSITÄT DARMSTADT

Two types of HTTP messages: request, response

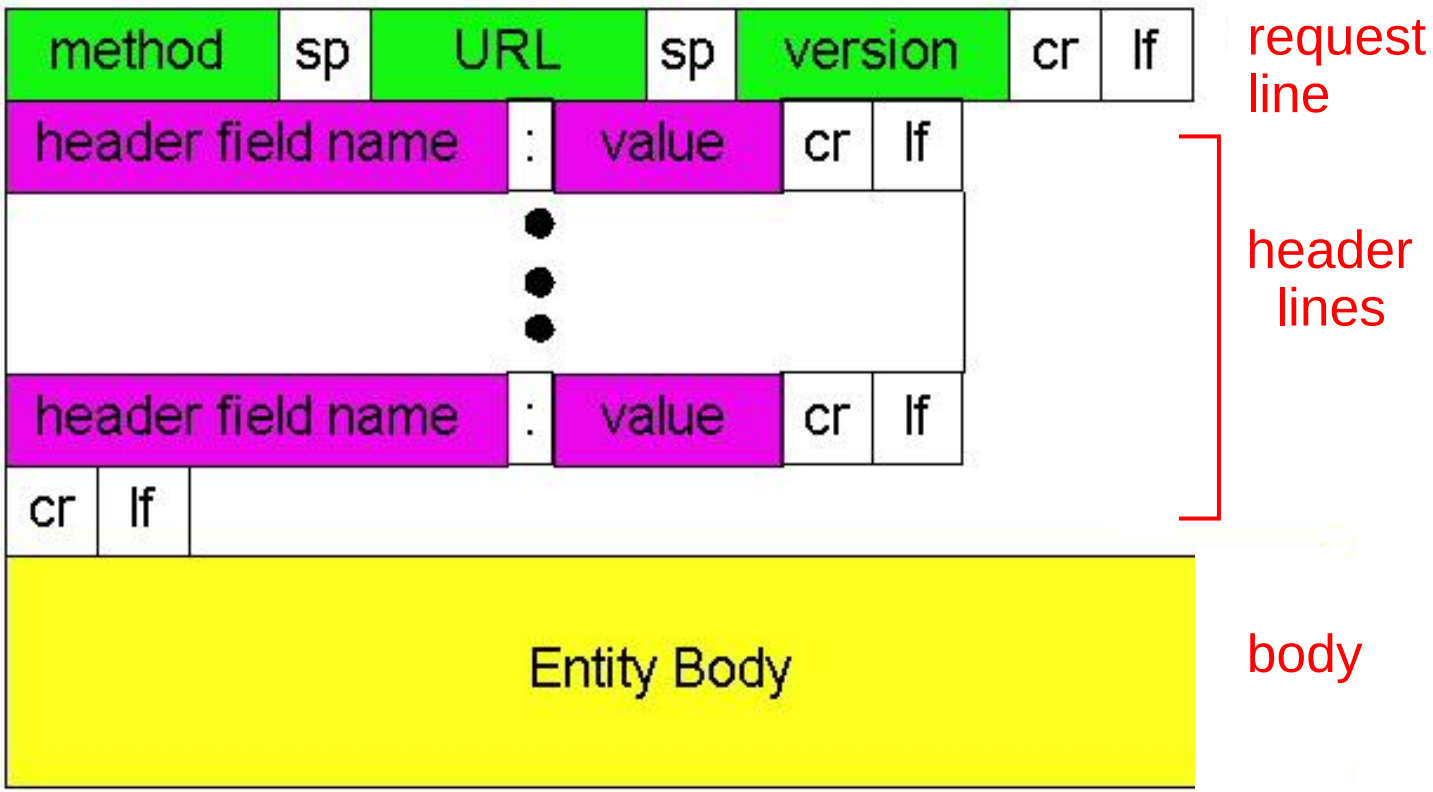
HTTP request message

- ASCII (human-readable format)



HTTP Request Messages

General format of HTTP requests



HTTP Request Messages

HTTP method types

HTTP/1.0

- GET
  - Request file specified in URL field
- POST
  - Send request with additional info in entity body
- HEAD
  - Asks server to leave requested object out of response
  - Response only is metadata contained in HTTP message header

HTTP/1.1

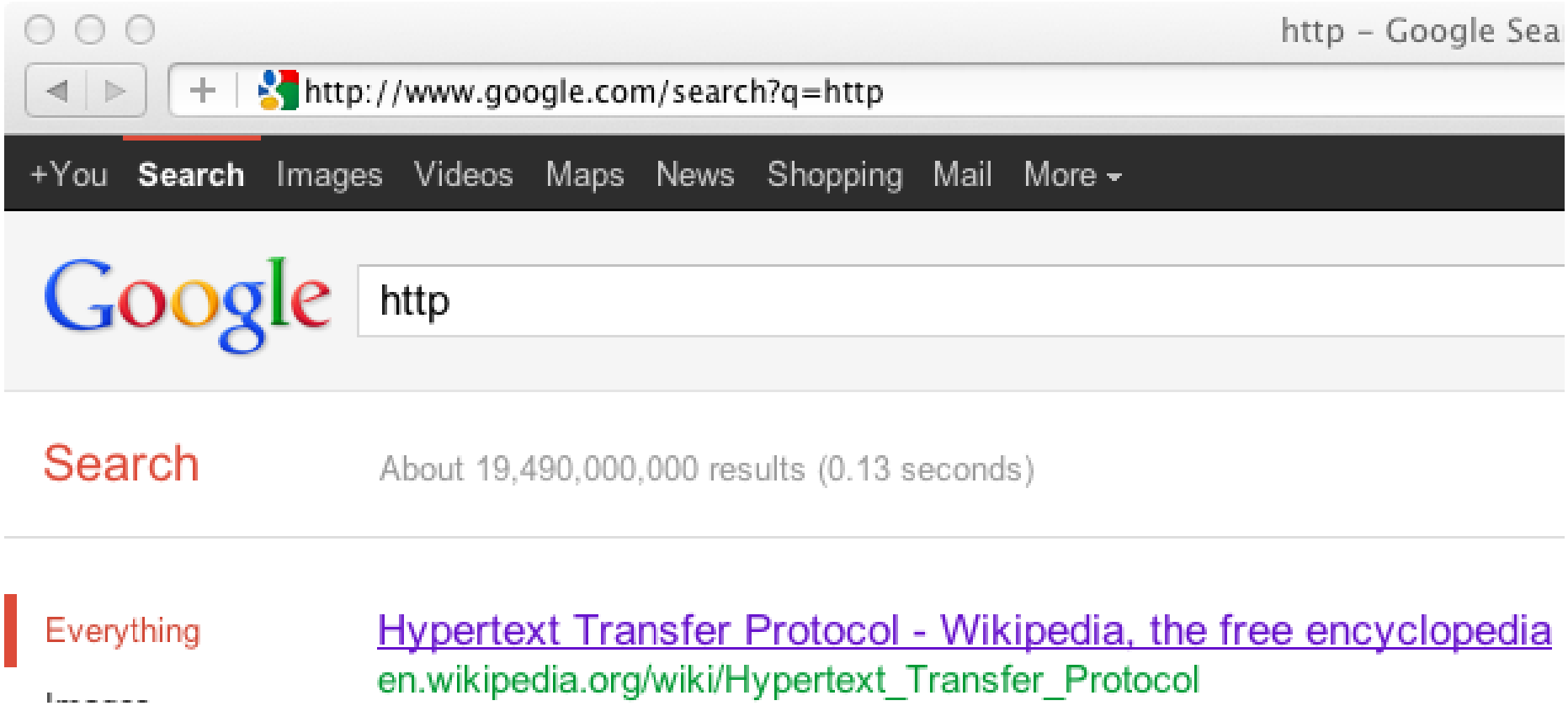
- GET, POST, HEAD
- PUT
  - Uploads file in entity body to path specified in URL field
- DELETE
  - Delete file specified in the URL field

HTTP Request Messages

HTTP knows two ways for uploading form input

1. form input based on GET method

- Input encoded in URL (query_string)
- Should be used for idempotent queries (i.e. al requests lead to same results)
  - I.e. reload is not harmful



TECHNISCHE UNIVERSITÄT DARMSTADT

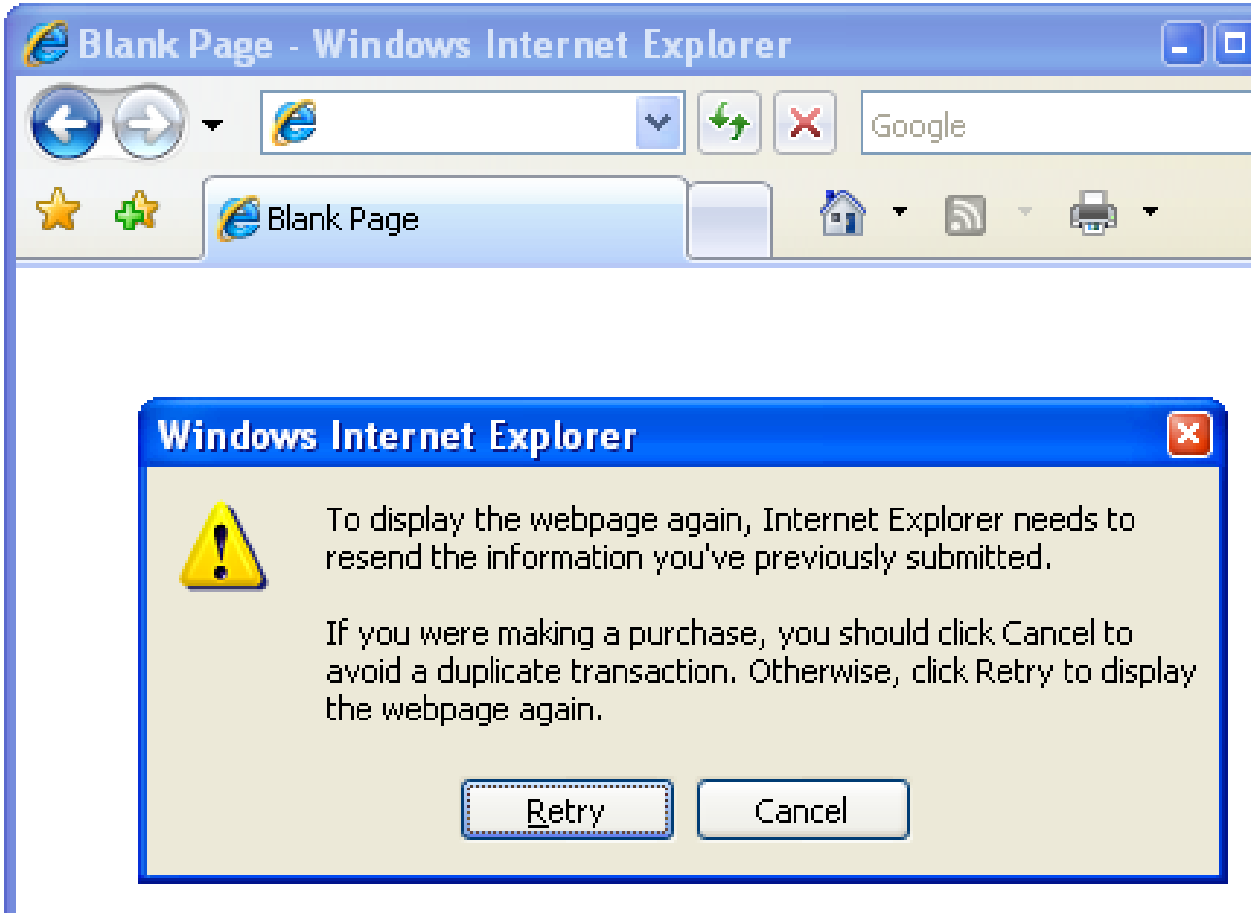
HTTP knows two ways for uploading form input (cont'd)

2. Form input based on POST method

- Input contained in entity body of HTTP request

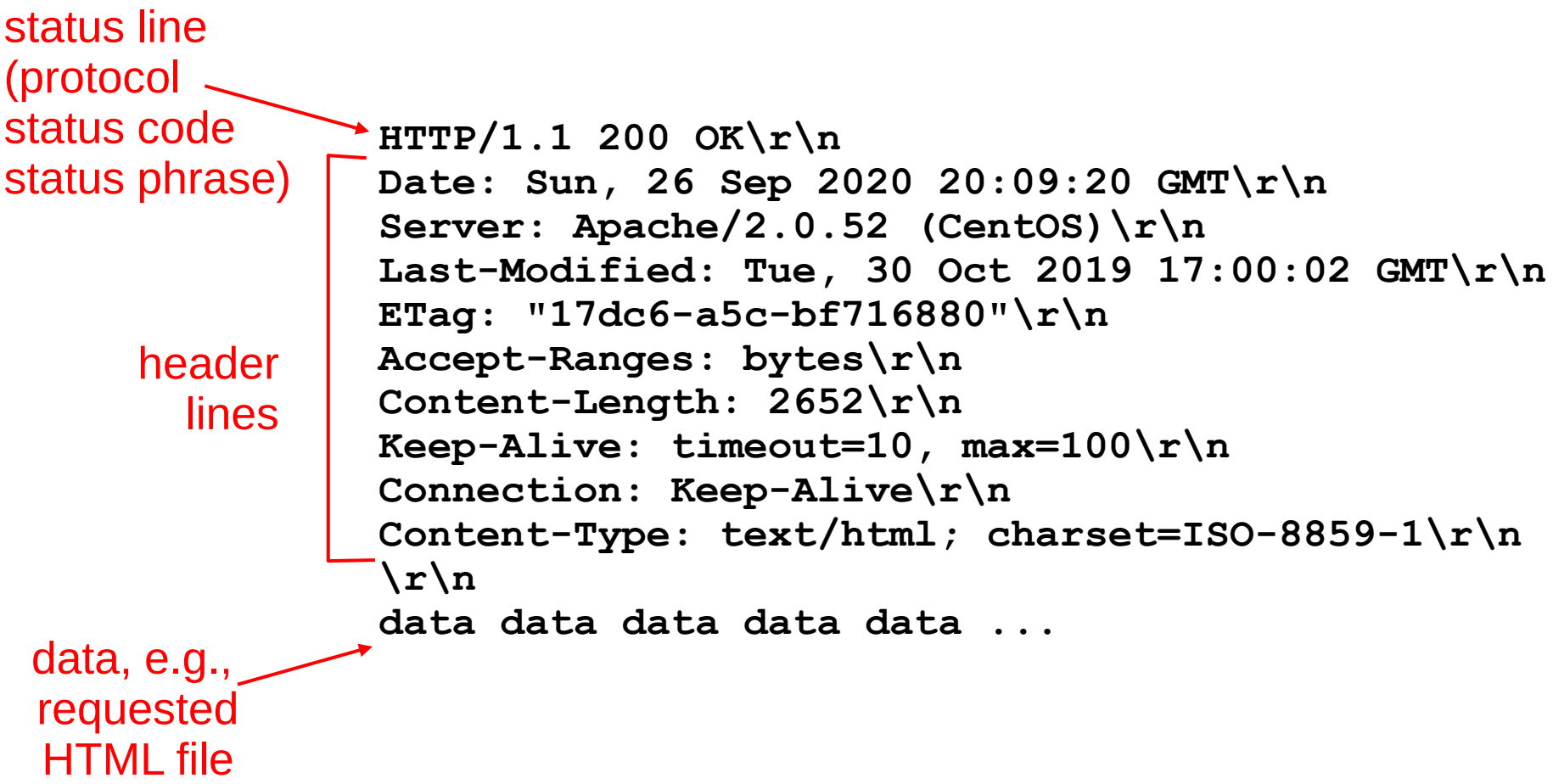
- Should be used for non-idempotent queries
  - I.e. reloading page would repeat actions (e.g. placing an order)
  - Browser will display a warning message

- Should be also used
  - If input is non ASCII
  - If input exceeds certain length



HTTP response message

- ASCII (human-readable format)



HTTP Response Messages

TECHNISCHE UNIVERSITÄT DARMSTADT

HTTP response status codes

- Appear in 1st line in server-client response message

Some sample codes

- 200 OK
  - Request succeeded, requested object contained in message
- 301 Moved Permanently
  - Requested object moved, new location specified in message
- 400 Bad Request
  - Request message not understood by server
- 404 Not Found
  - Requested document not found on this server
- 505 HTTP Version Not Supported

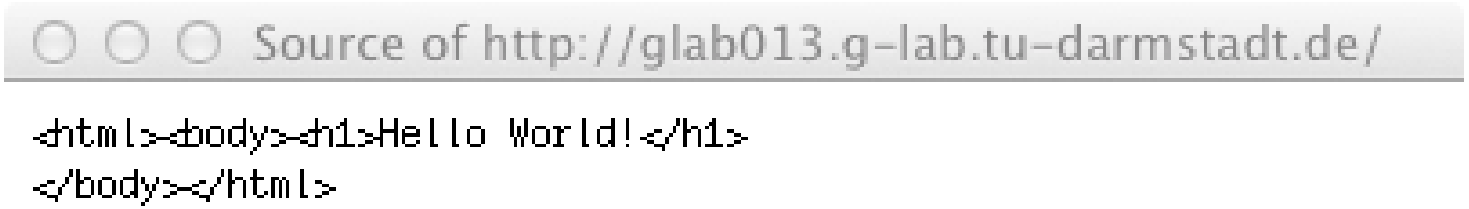
TECHNISCHE UNIVERSITÄT DARMSTADT

Example: accessing simple web page

- File index.html at server glab013.g-lab.tu-darmstadt.de

Hello World!

- Source code received by client's web browser



HTTP Example

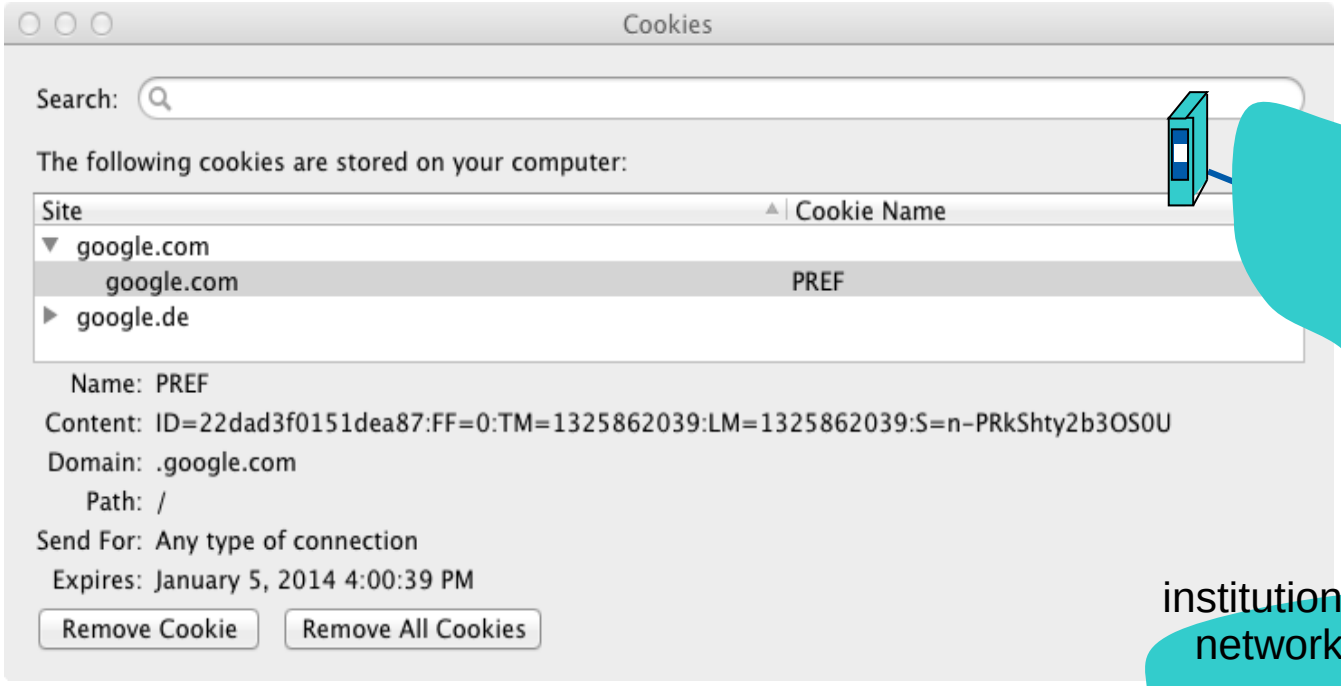
Example: accessing simple web page (cont'd)

- Talking HTTP using Telnet

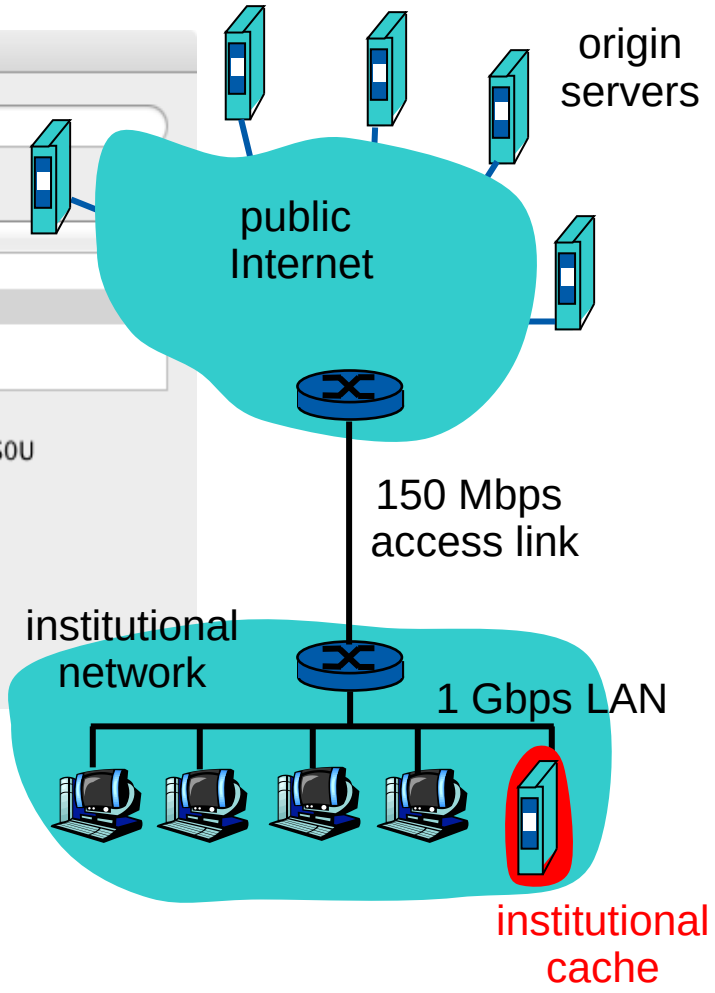
```
dhcp47:~ Andre$ telnet glab013.g-lab.tu-darmstadt.de 80
Trying 130.83.125.13...
Connected to glab013.g-lab.tu-darmstadt.de.
Escape character is '^]'.
GET /index.html HTTP/1.1
Host: glab013.g-lab.tu-darmstadt.de

HTTP/1.1 200 OK
Date: Fri, 06 Jan 2012 14:21:23 GMT
Server: Apache/2.2.20 (Ubuntu)
Last-Modified: Fri, 06 Jan 2012 13:41:40 GMT
ETag: "19e6-31-4b5dc356b8f8e"
Accept-Ranges: bytes
Content-Length: 49
Vary: Accept-Encoding
Content-Type: text/html

<html><body><h1>Hello World!</h1>
</body></html>
Connection closed by foreign host.
dhcp47:~ Andre$
```

TECHNISCHE UNIVERSITÄT DARMSTADT

Cookies



Web Caches



TECHNISCHE UNIVERSITÄT DARMSTADT

Cookies allow to maintain user state on server

- Recall: HTTP itself is stateless protocol
- Latest specification in RFC 6265 as of April 2011
  - First specification in RFC 2109 as of February 1997
- Used to remember user-specific server settings
  - Authorization, shopping carts, recommendations, session state
- Used by many web sites today
  - Take a look at the cookie list of your browser…

Four components required

- Cookie header line of HTTP response message
  - To transfer cookie from server to client
- Cookie header line of HTTP request message
  - To send back cookie form client to server
- Cookie file managed by client's browser
  - General format: name:value pairs
- Back-end database at web server
  - Storing user state

Cookies

TECHNISCHE UNIVERSITÄT DARMSTADT

Cookies allow to maintain user state on server

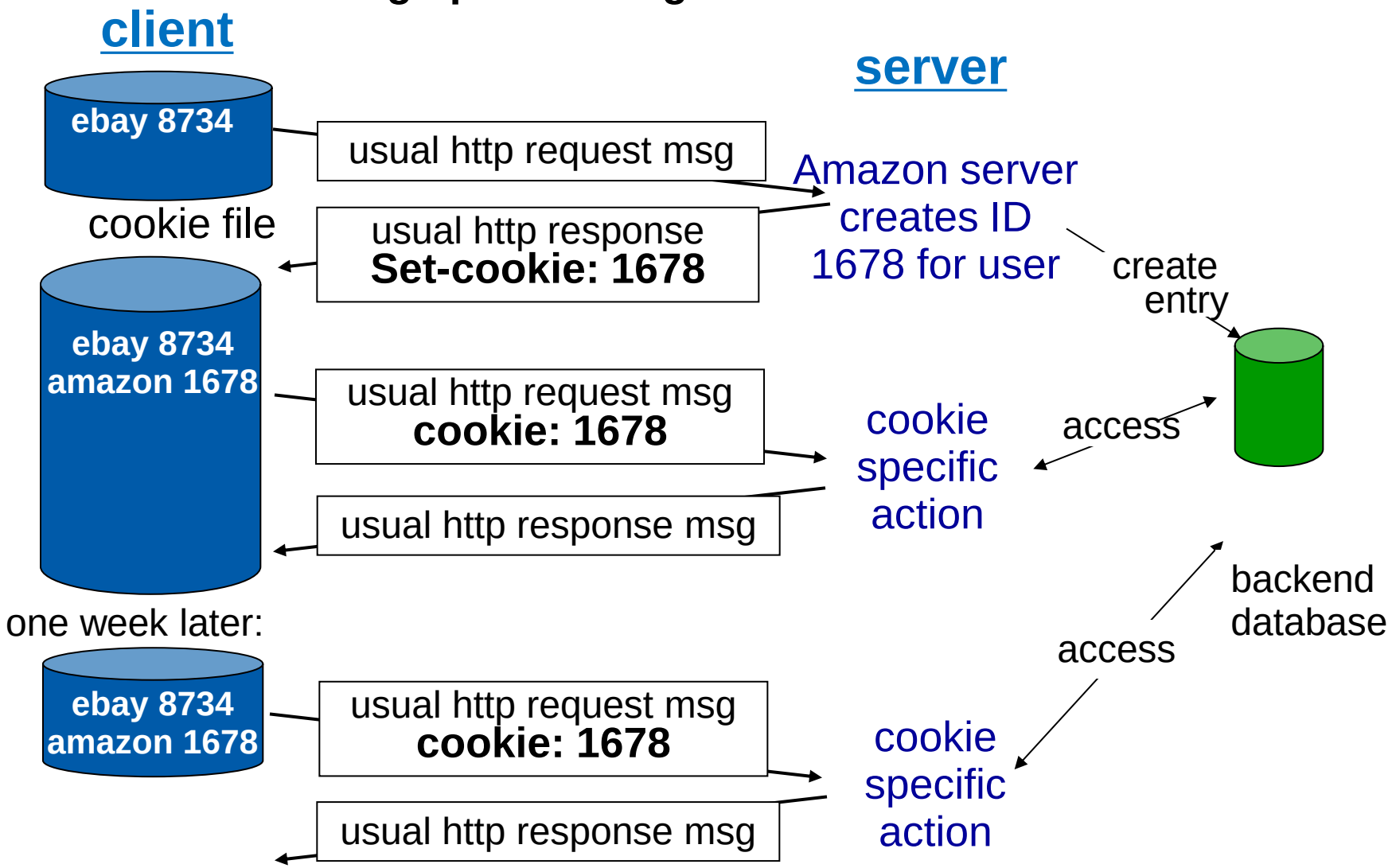
- Recall:
  HTTP itself is stateless protocol
- Latest specification in RFC 6265 as of April 2011
  - First specification in RFC 2109 as of February 1997
- Used to remember user-specific server settings
  - Authorization, shopping carts, recommendations, session state
- Used by many web sites today
  - Take a look at the cookie list of your browser…

Four components required

- Cookie header line of HTTP response message
  - To transfer cookie from server to client
- Cookie header line of HTTP request message
  - To send back cookie form client to server
- Cookie file managed by client's browser
  - General format: name:value pairs
- Back-end database at web server
  - Storing user state

TECHNISCHE UNIVERSITÄT DARMSTADT

Setting up and using cookies



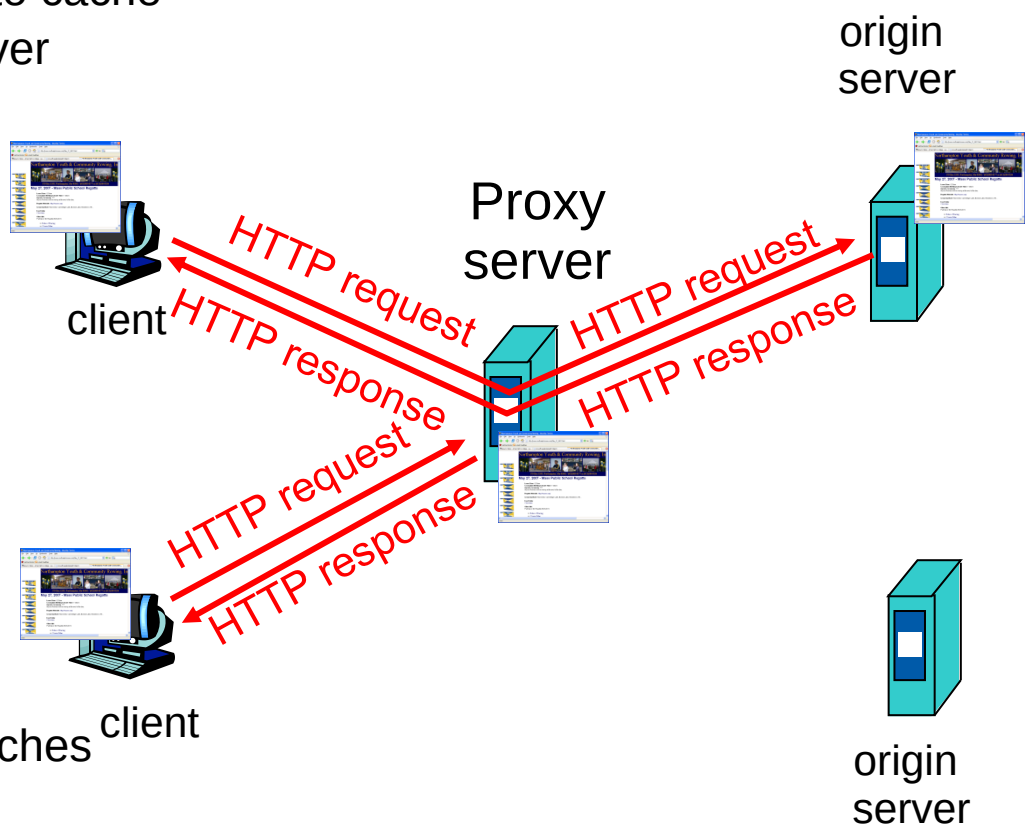
TECHNISCHE UNIVERSITÄT DARMSTADT

Example: Firefox cookies

- Settings - privacy

Goal: to satisfy client request without involving origin server

- Typically cache is installed by ISP
  - University, company, residential ISP
- User sets in browser to access web via cache
- Browser sends all HTTP requests to cache
- Cache acts as both, client and server
- If Object in cache
  - Cache returns object
  - Else cache requests object
    - From origin server
    - Then returns object to client



Why web caching?

- To reduce response time for client request
- To reduce traffic on an institution's access link
- To increase Internet dense with caches
  - Enables "poor" content providers to effectively deliver content

Web Caches (Proxy Servers) - Example

Problem Statement

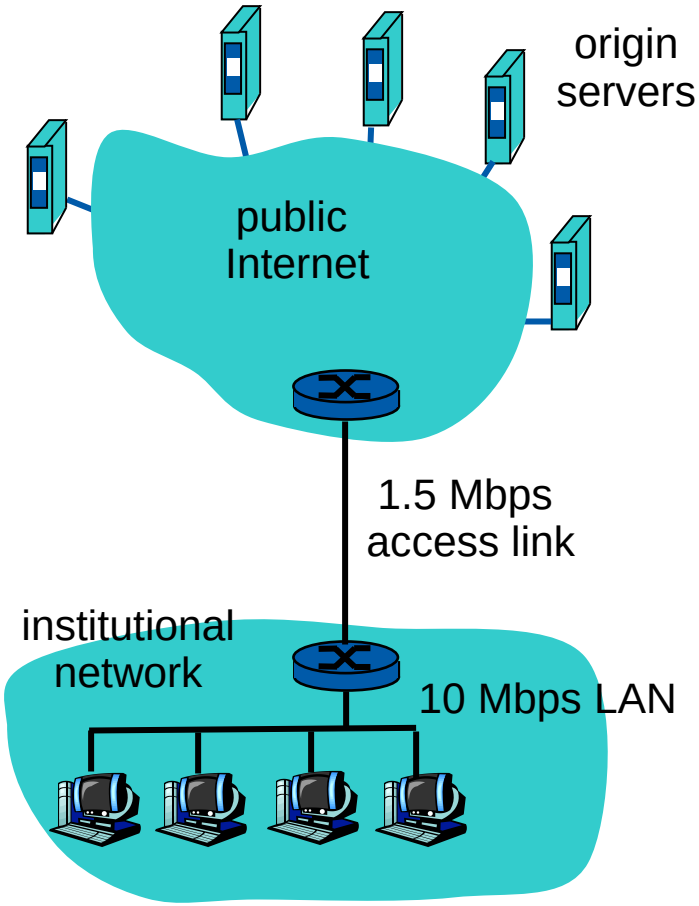
Assumptions

- Average object size = 100.000 bits
- Average request rate from institution's browsers to origin servers = 15/sec
- Delay from institutional router to any origin server and back to router = 2 sec

Consequences

- Utilization on LAN = 15%
- Utilization on access link = 100%

- Total average delay
  = Internet delay + access delay + LAN delay
  = 2 sec + minutes + milliseconds



Web Caches (Proxy Servers) - Example

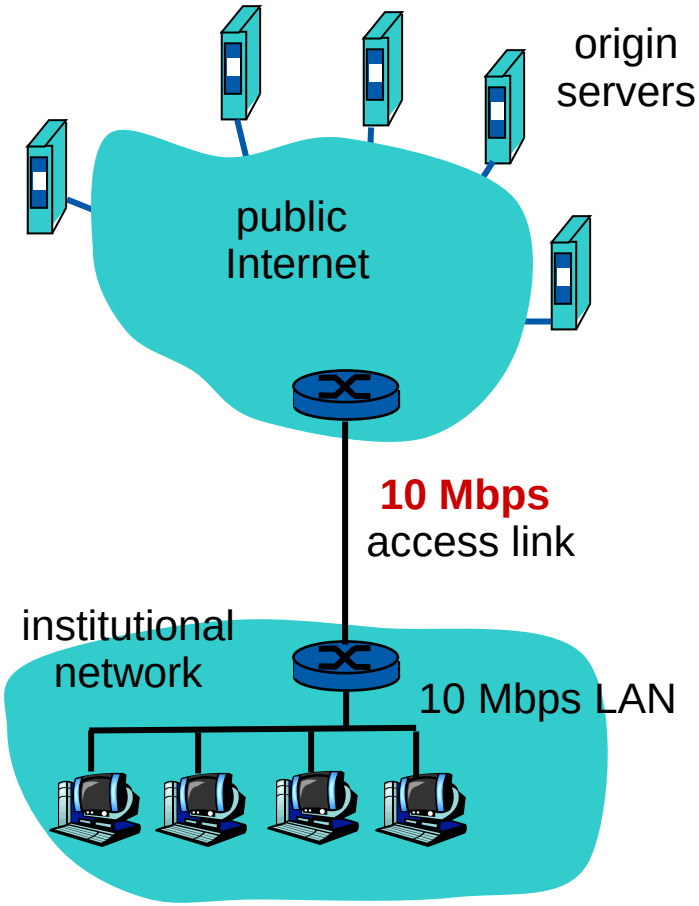
Possible 1.st solution

- Increase bandwidth of access link to e.g. 10 Mbps

Consequences

- Utilization on LAN = 15%
- Utilization on access link = 15%

- Total average delay
  = Internet delay + access delay + LAN delay
  = 2 sec + milliseconds + milliseconds
- But:
  - often a costly upgrade



Web Caches (Proxy Servers) - Example

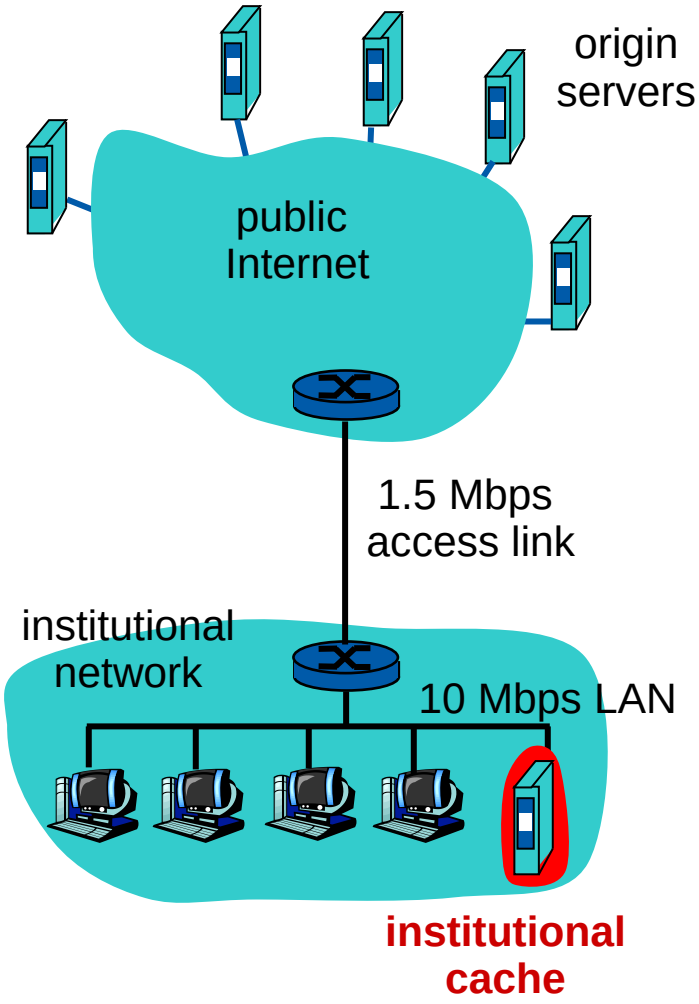
Possible cache solution

- Install cache

Consequence

- Suppose hit rate is 0.4
  - 40% requests satisfied almost immediately
  - 60% requests satisfied by origin server
- Utilization of access link reduced to 60%
  - Resulting in negligible delays (e.g. 10 msec)

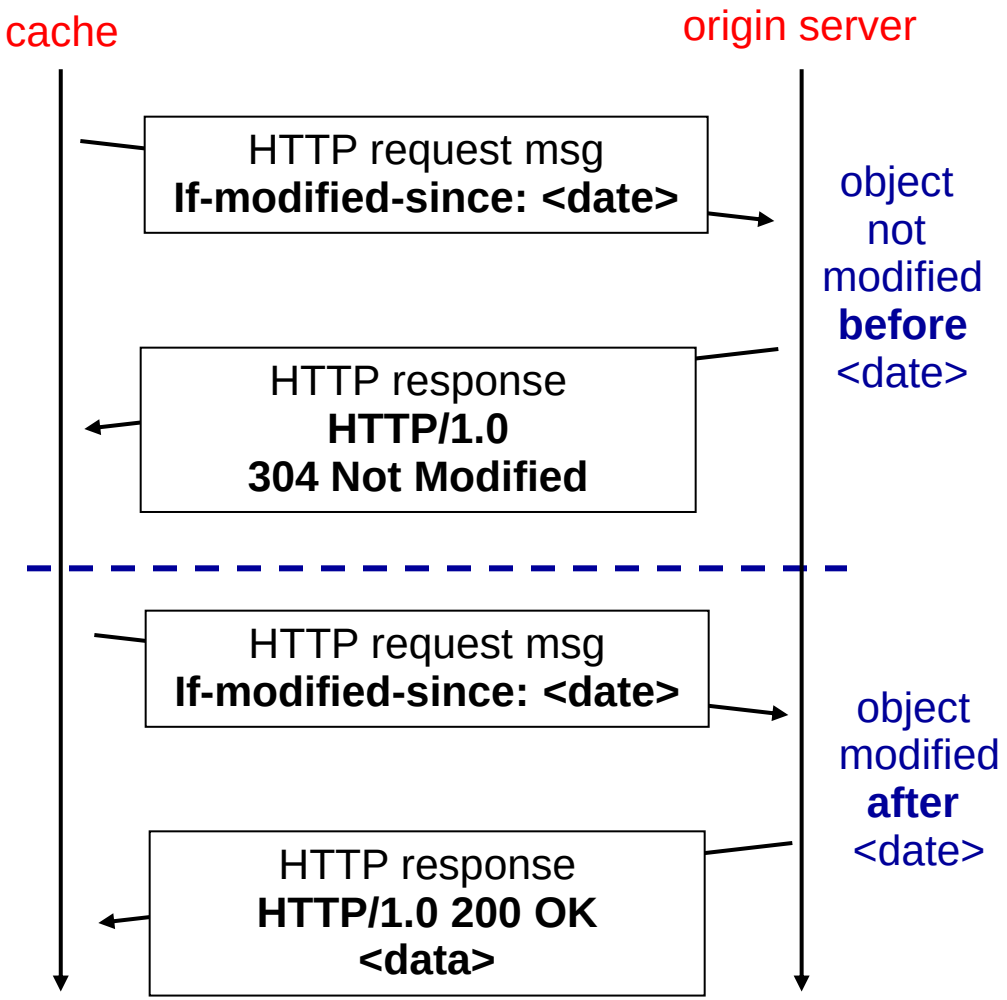
- Total average delay
  = Internet delay + access delay + LAN delay
  = 0.6*(2.01) secs + 0.4*milliseconds
  < 1.4 secs



Web Caches (Proxy Servers) - Example

Conditional GET

- Goal:
  do not send object
  if cache has up-to-date
  cached version
- Cache:
  specify date of
  cached copy in HTTP request
  - If-modified-since: <date>
- Server:
  response contains
  no object if cached copy
  is up-to-date
  - HTTP/1.0 304 Not Modified



Internet hosts, routers

- Addressed by machine readable IP address
  - Used for routing packets
- Addressed also by human readable “Name”
  - E.g. www.google.com
- How to map between IP address and name, and vice versa?
- At the early days
  hosts.txt file was updated and transferred manually…

Domain Name System

- Distributed database
  - Implemented in hierarchy of many name servers
- Application-layer protocol
  - Used by host, routers, name servers
  - To communicate and resolve names (address/name translation)
- Note: DNS is one of Internet’s core functions
  - Implemented as application-layer protocol
  - Thus complexity at network’s “edge”

Domain Name System (DNS)

TECHNISCHE UNIVERSITÄT DARMSTADT

DNS services

- Hostname to IP address translation
- Host aliasing
  - Canonical, alias names
- Mail server aliasing
- Load distribution
  - E.g. replicated web servers
    - Set of IP addresses for one canonical name

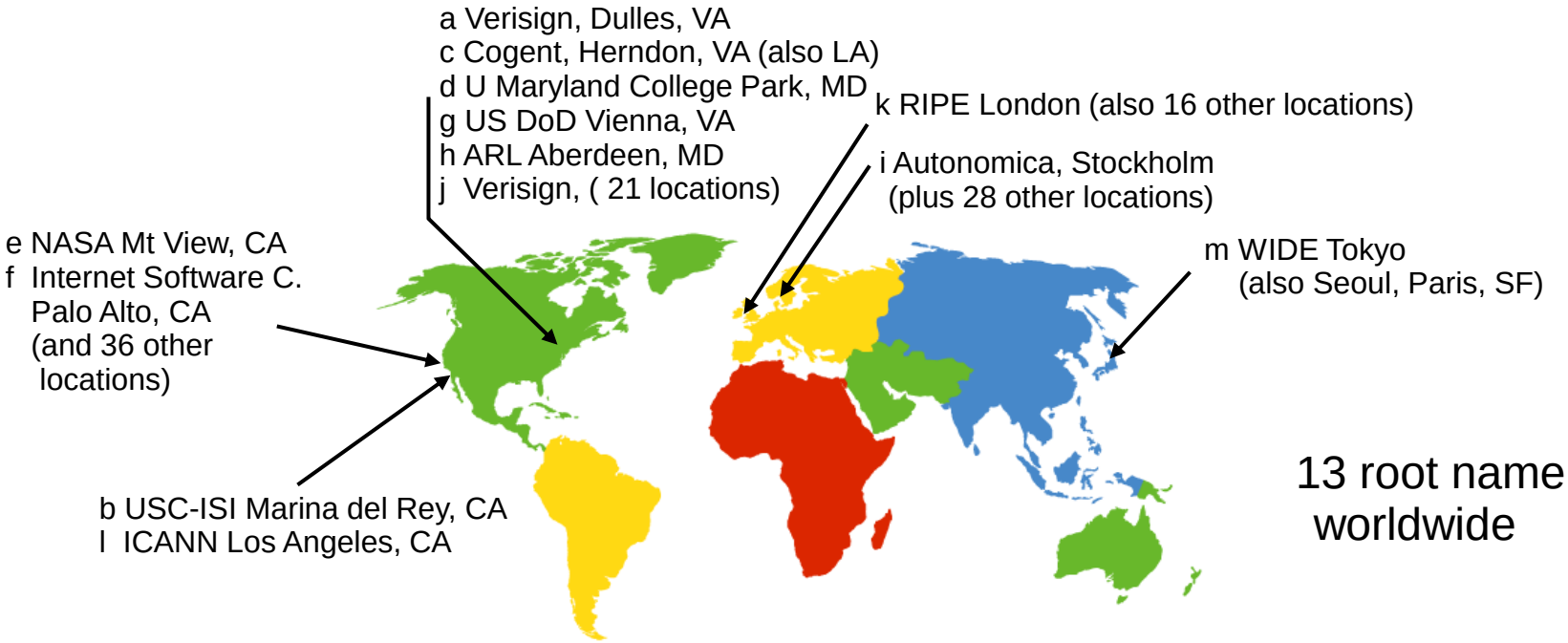
Why not centralize DNS?

- Single point of failure
- Traffic volume
- Distant centralized database
- Maintenance
  → Does not scale!

TECHNISCHE UNIVERSITÄT DARMSTADT

DNS root name servers

- Contacted by local name server that can not resolve name
- Root name server
  - Contacts authoritative name server
    - If name mapping not known
  - Gets mapping
  - Returns mapping to local name server



13 root name servers worldwide

DNS Structure

TECHNISCHE UNIVERSITÄT DARMSTADT

Top-level domain (TLD) servers

- Responsible for
  - .com, .org, .net, .edu, .aero, .jobs, .museums
  - And all top-level country domains, e.g.: de, uk, fr, ca, jp
- Network Solutions maintains servers for .com TLD
- Educause maintains servers for .edu TLD

Authoritative DNS servers

- Organization's DNS servers
  - Providing authoritative hostname to IP mappings for organization's servers (e.g., web, mail)
- Can be maintained by organization or service provider

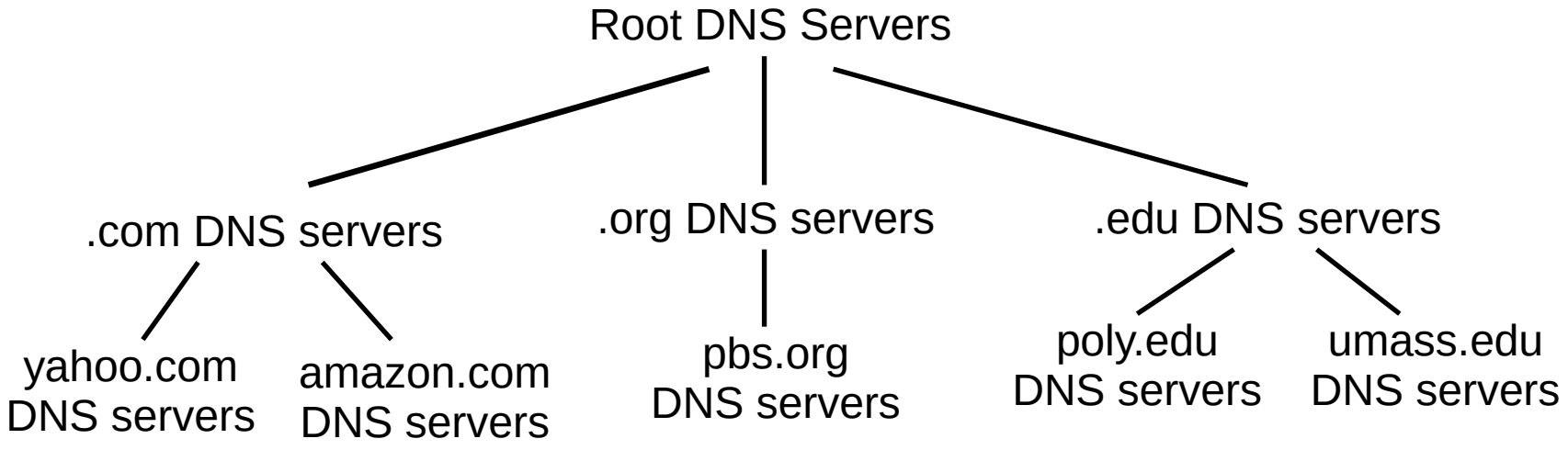
Local name server

- Does not strictly belong to hierarchy
- Each ISP (residential, company, university, …) has one
  - Also called default name server
- End system's DNS queries usually sent to local DNS server
  - Acts as proxy, forwards query into hierarchy

DNS Structure

Client wants IP for www.amazon.com

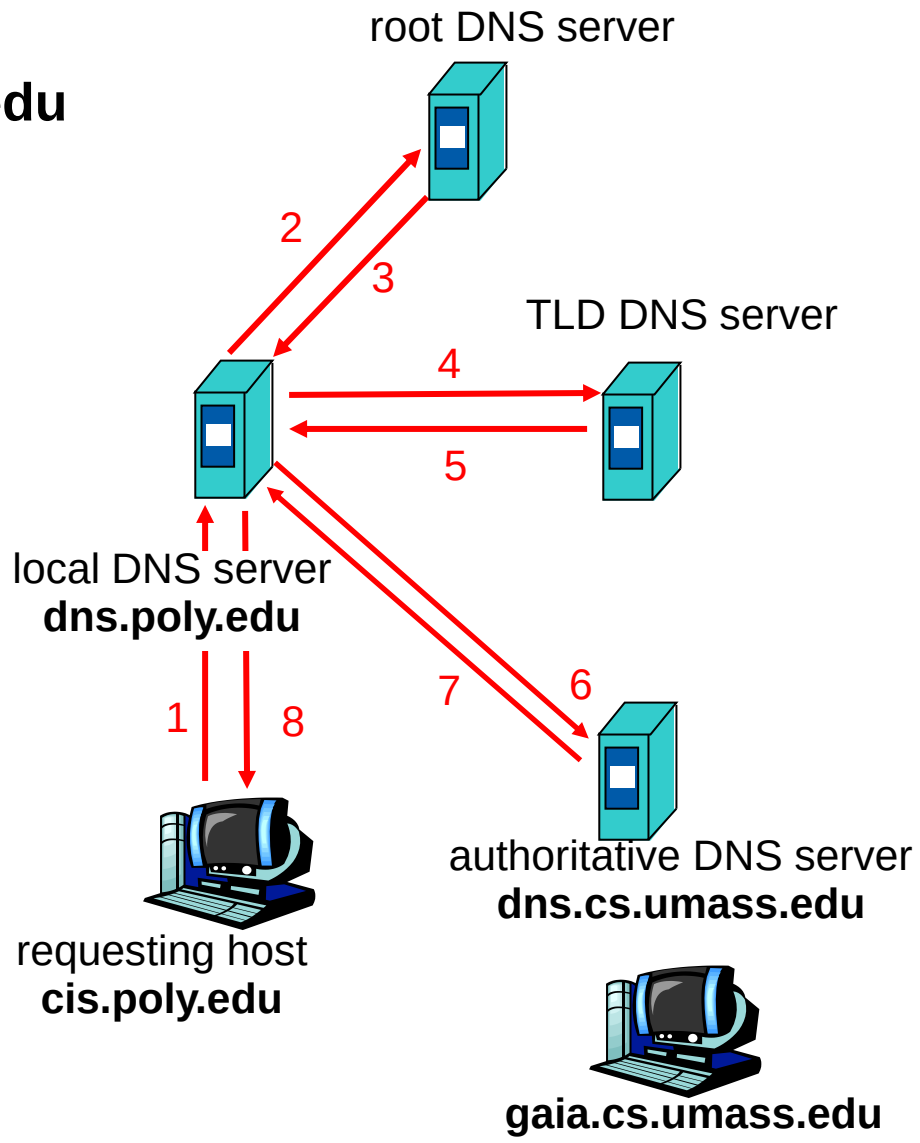
- First approximation
  - Client queries a root server
    - To find .com DNS server
  - Client queries .com DNS server
    - To get amazon.com DNS server
  - Client queries amazon.com DNS server
    - To get IP address for www.amazon.com



Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

Iterative query

- Local name server contacts server in hierarchy iteratively
- Contacted server replies
  - With name of server to contact
- "I don't know this name, but ask this server"

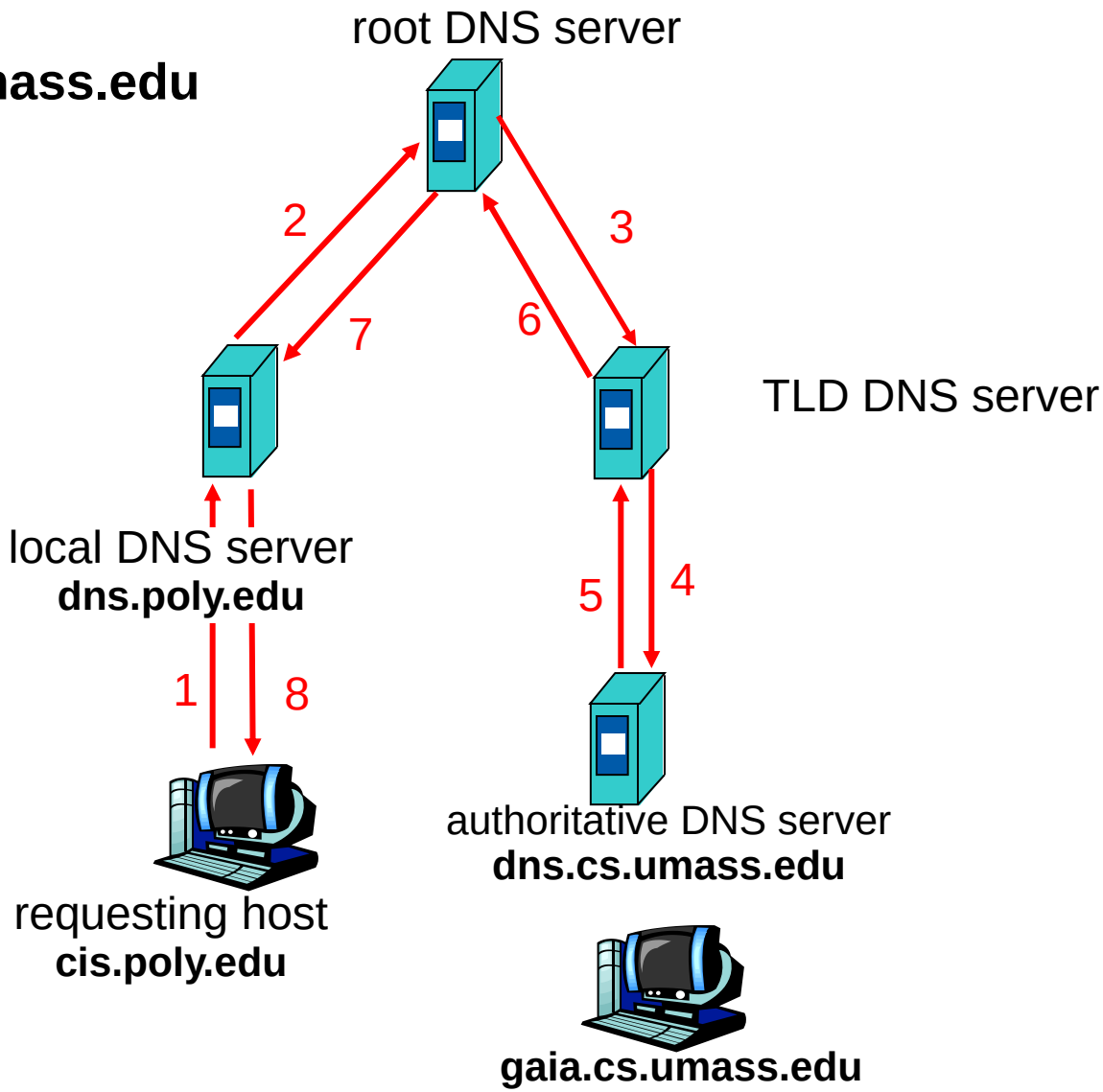


DNS Name Resolution Example

Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

Recursive query

- Each name server involved contacts next server in hierarchy
- Puts burden of name resolution on contacted name server
- Heavy load?



TECHNISCHE UNIVERSITÄT DARMSTADT

DNS's distributed database stores resource records (RR)

- Format: (*name, value, type, ttl*)

- Type A (authoritative)
  - *name* is hostname
  - *value* is IP address

- Type AAAA
  - Same as A for IPv6

- Type NS (name server)
  - *name* is domain (e.g. tu-darmstadt.de)
  - *value* is hostname of authoritative name server of domain

- Type CNAME (canonical name)
  - *name* is alias for real name
  - *value* is canonical name
  - E.g. www.tu-darmstadt.de is really vwcms-live01.hrz.tu-darmstadt.de

- Type MX (mail exchanger)
  - *value* is name of mail server associated with *name*

Records are cached and updated

- Once a name server learns a mapping it caches it
- Cache entries time out after some time
- TLD servers typically cached in local name servers
  - Thus root name servers not often visited
- Update/notify mechanisms defined April 1997 in RFC 2136

TECHNISCHE UNIVERSITÄT DARMSTADT

Inserting DNS records

- Example: new startup “Company”
- Registers name company.com at DNS registrar
  - Accredited by Internet Corporation for Assigned Names and Numbers ICANN
  - In Germany e.g. Deutsche Telekom, 1&1
- Provides names and IP addresses of authoritative name server

- Registrar admin inserts two resource records into .com TLD server
  - (company.com, dns1.company.com, NS)
  - (dns1.company.com, 123.234.345.1, A)

- company.com admin creates at authoritative name server
  - Type A record for www.company.com
  - Type MX record for company.com

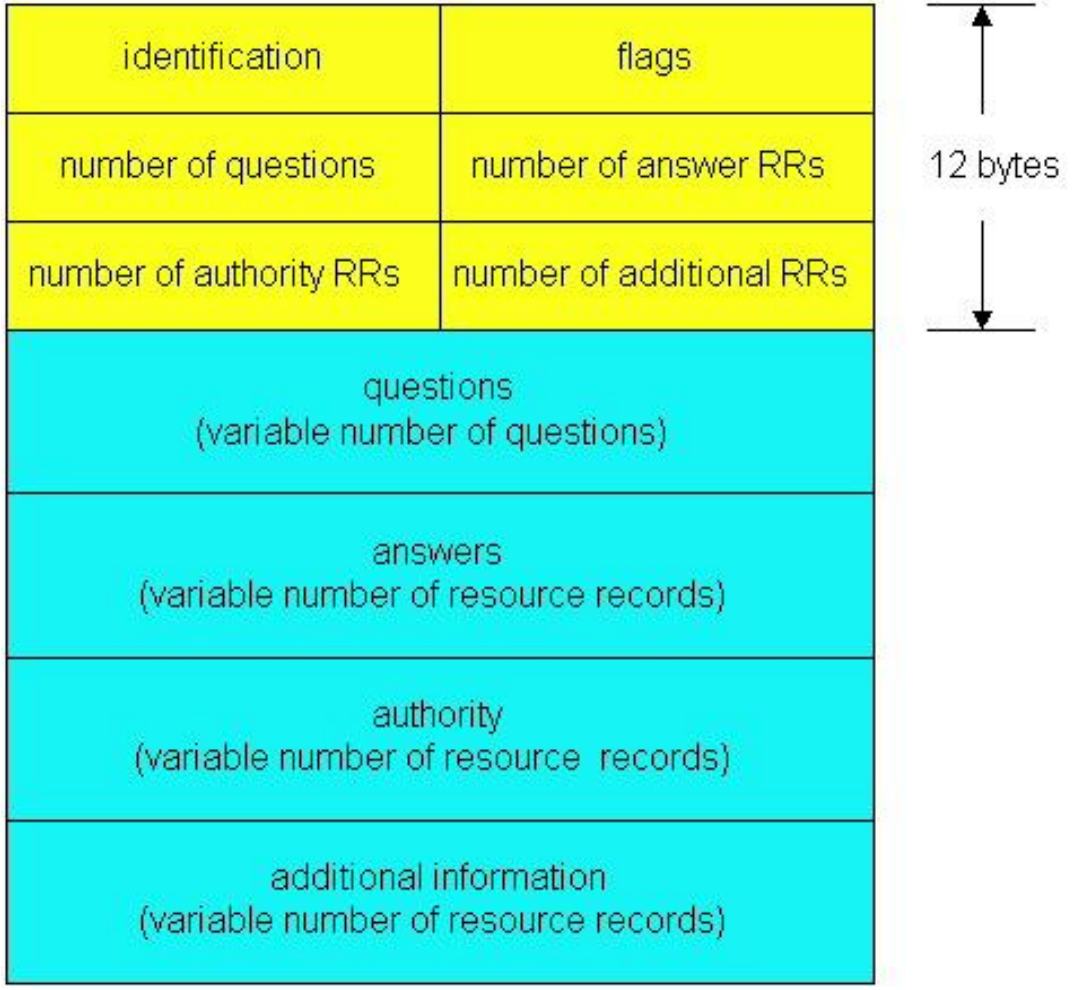
TECHNISCHE UNIVERSITÄT DARMSTADT

DNS protocol specifies query and reply messages

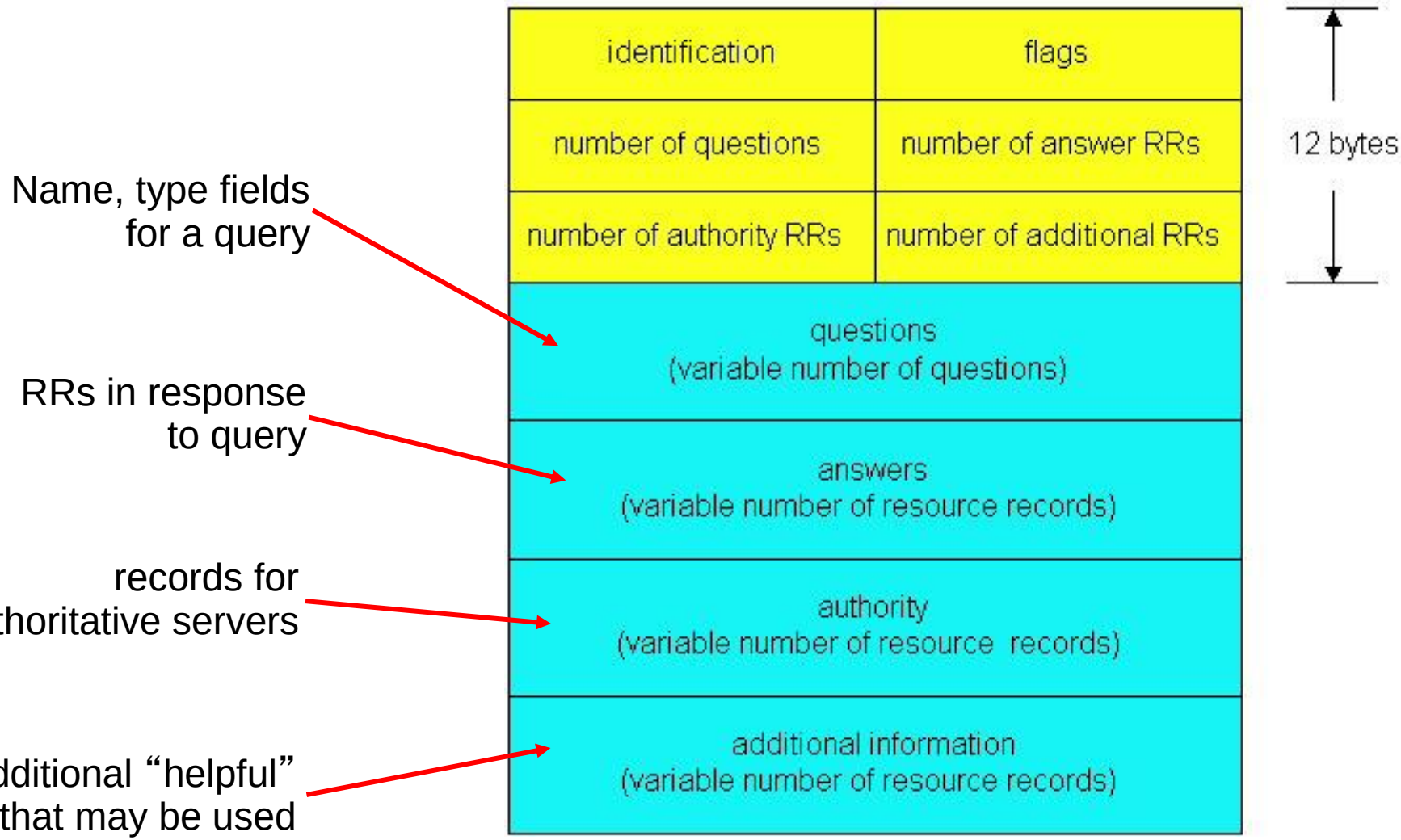
- Both with same message format

Message header

- Identification
  - 16 bit number generated for query
  - Reply to query uses same number
- Flags
  - Query or reply
  - Recursion desired
  - Recursion available
  - Reply is authoritative



DNS Protocol



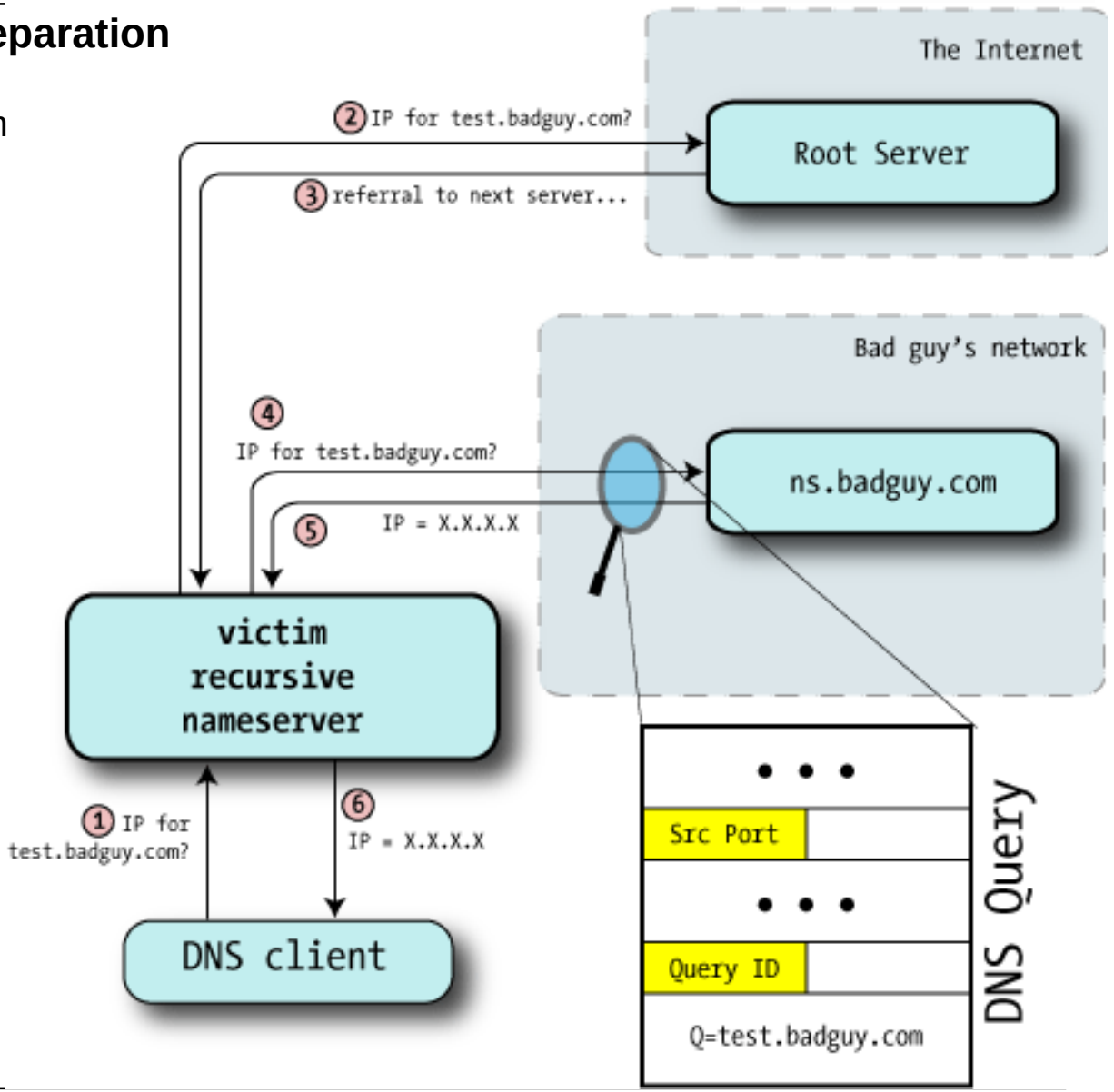
DNS information is not authenticated

- In early days poisoning DNS caches was very easy
  - Attacker makes victim query for www.attacker.com
    - Query sent to attacker's name server (e.g. ns.attacker.com)
  - Attacker's name server replied with A record for www.attacker.com
    - Adds additional information (e.g. A record for www.bank-of-victim.com)
    - Victim trusts additional information
    - Victim's access to www.bank-of-victim.com redirected
- Today DNS implementations are more paranoid
  - E.g. additional information for foreign domains not accepted
- Injecting bogus ("malicious") information more complicated but still possible
- Authenticated DNS (DNSSEC) is yet to come
  - Established for root zone mid 2010
    - I.e. records for most top level domains can be validated

DNS Security Issues

DNS poisoning example - preparation

- Attacker wants to poison DNS entry for www.bankofsteve.com
- Challenge 1: DNS server only accepts authoritative replies
  - I.e. attacker has to blindly reply to legitimate request sent to authoritative name server of domain bankofsteve.com
- Challenge 2: DNS server only accepts replies with correct query ID
  - I.e. attacker has to guess query ID used
  - Assumption: query IDs are incremented linearly
  - Approach: learn query ID by forcing victim name server to send request to attackers name server

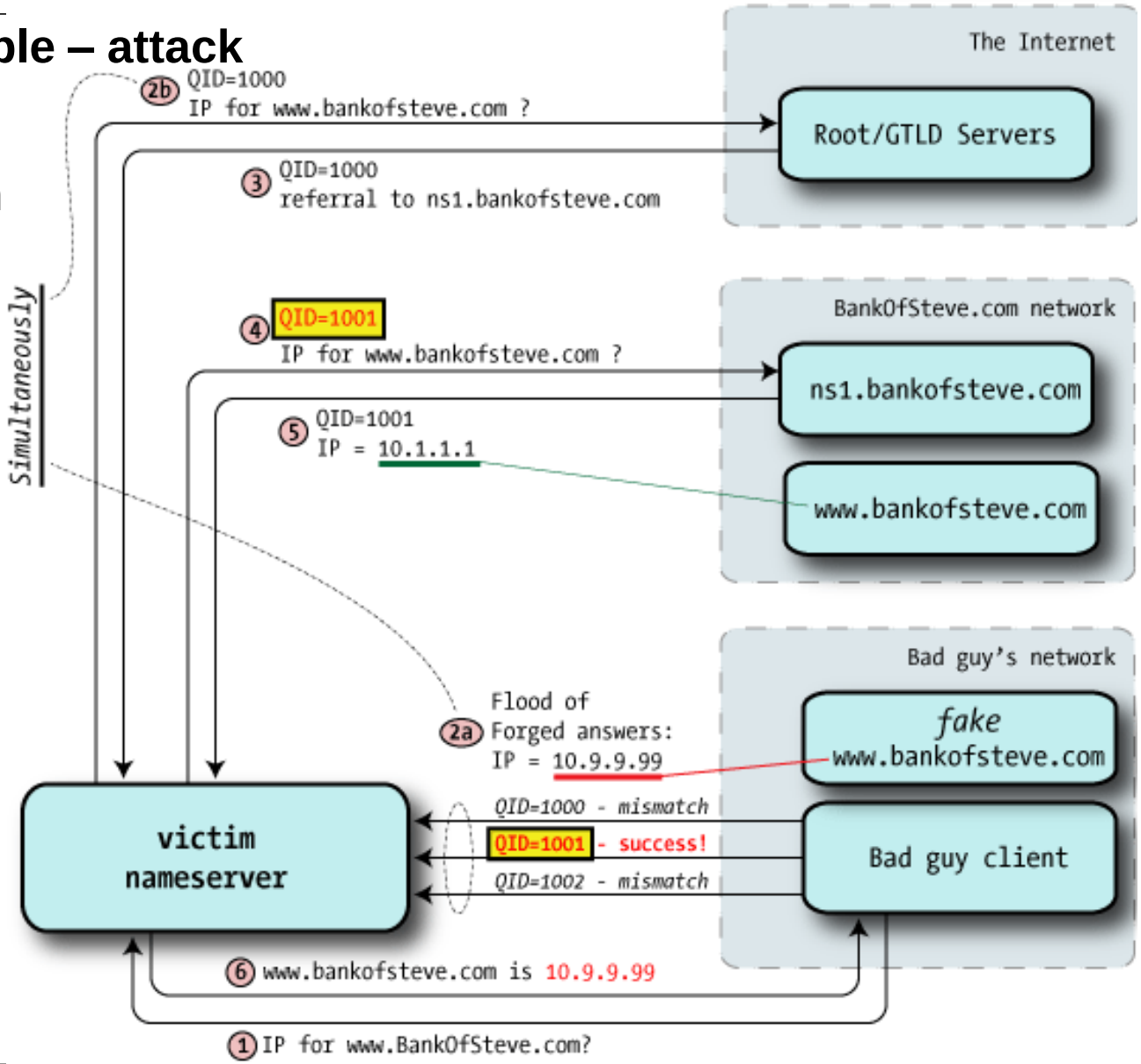


Source: http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

TECHNISCHE UNIVERSITÄT DARMSTADT

DNS poisoning example – attack

- Attacker forces victim name server to query www.bankofsteve.com
- Attacker floods victim name server with faked replies
  - Using guessed query IDs
- Attacker succeeds if
  - Query ID guessed correctly
  - Reply arrives before legitimate reply



Source: http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

DNS poisoning – amplification

- So far: poisoned A record
  - One DNS reply will contain faked IP address
- One step further: poisoned NS record
  - All DNS requests will be handled by attacker's name server
  - Presented by Dan Kaminsky in 2008

DNS poisoning and https

- https uses digital certificates to uniquely bind IP address to URL
- But: www.bankofluxemburg.com vs. www.bankofiuxemburg.com …

DNS poisoning – countermeasures

- Randomize query IDs to prevent guessing
  - But experiments showed chances are still good for 216=65536 possible IDs
- Randomize query IDs and UDP port used to send query
  - But port translating NAT devices may nullify port randomization

TECHNISCHE UNIVERSITÄT DARMSTADT

Recalling Licklider's visions

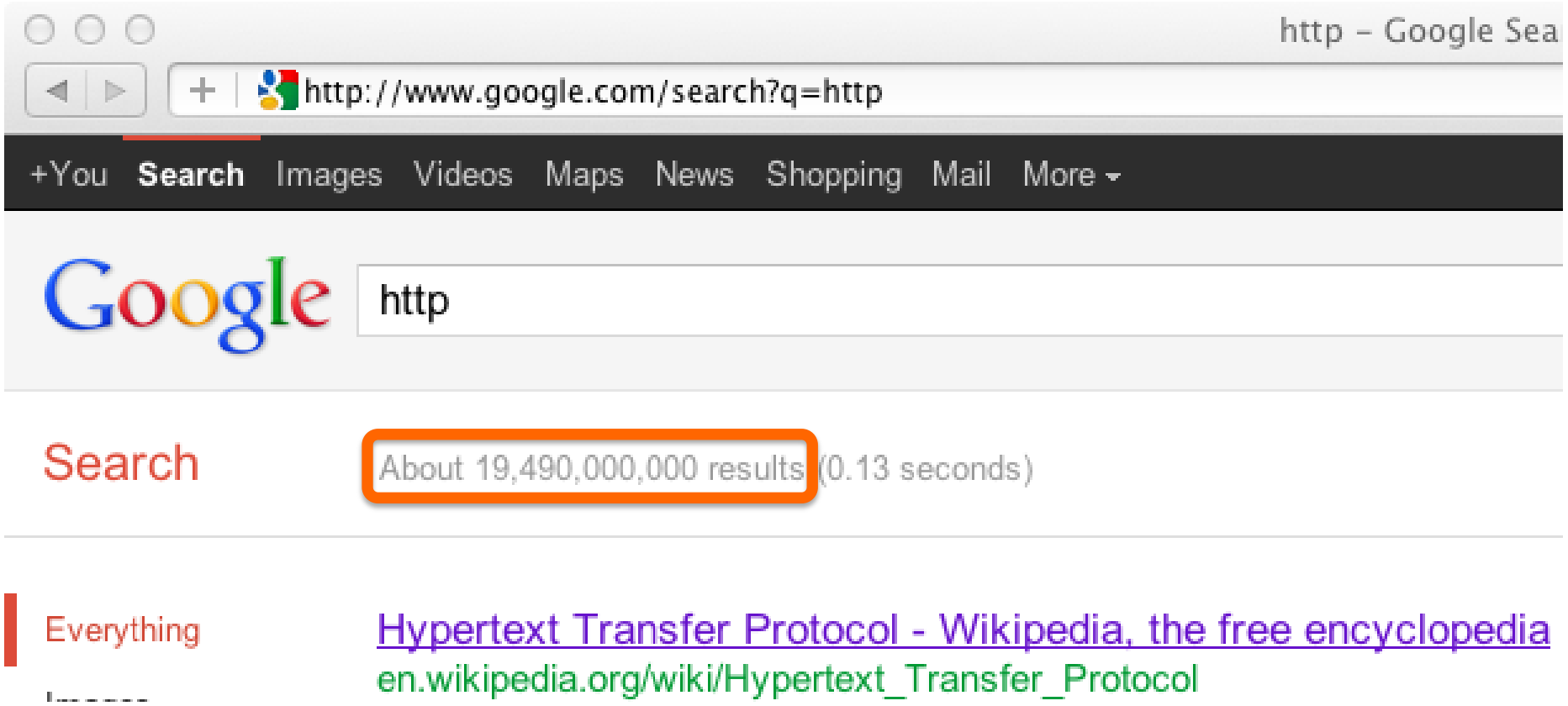
Obviously, collections of primary data can get too large to digest. There comes a time when the complexity of a communications process exceeds the available resources and the capability to cope with it; and at that point one has to simplify and draw conclusions.

Source: The Computer as a Communication Device, Science and Technology, 1968, reprint

TECHNISCHE UNIVERSITÄT DARMSTADT

Non-semantic web with a googol of pages

- Which are the relevant ones?
- Which are proved to be good?
- Which are still accessible?



6.2 Semantic Web

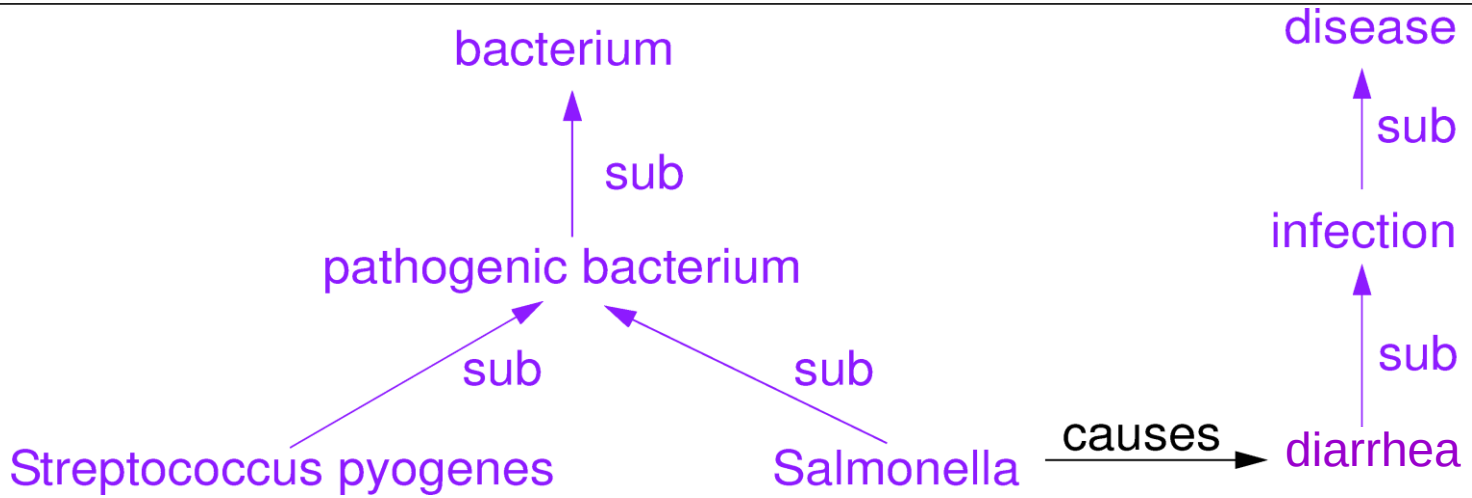
“The Semantic Web
provides a common framework that allows DATA TO BE SHARED AND REUSED across application, enterprise, and community boundaries. It is a COLLABORATIVE EFFORT LED BY W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URLs for naming.”

http://www.w3.org/2001/sw/
http://www.semanticweb.org/

”The Semantic Web
is an extension of the current web in which INFORMATION IS GIVEN WELL-DEFINED MEANING, better enabling computers and people to work in cooperation.”

Tim Berners-Lee, James Hendler, Ora Lassila, The Semantic Web, Scientific American, May 2001
http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html

Semantics – Ontologies



Metadata = data about data

- Defined in a metadata scheme
- Clear vocabulary for entries

Ontology = a specification of a conceptualization,
i.e. a shared conceptualization of a knowledge domain (Gruber)

- Concepts, instances as thematic entities of the knowledge domain
- Relations as semantic interconnections between concepts
  - Superconcepts - subconcepts, domain relations
  - Moreover: axioms, attributes, inference

Adding Semantics to the Web

TECHNISCHE UNIVERSITÄT DARMSTADT

E.g. semantic web as overlay on top of existing web resources