CONTENT of this CHAPTER

- DNS
- HTTP and WWW
- EMAIL
- SNMP

WWW and HTTP: Basic Concepts

- With a **browser** you can request for remote resource (e.g. an HTML file)
- **Web server** replies to queries (e.g. response containing HTML file)
- The protocol used for this type of communication is **HTTP**

- HTML file typically contain references to other servers/ressources
  - hyperlink with uniform resource locators (URL)
  - e.g. http://www.name.com/index.html
- You obtain pointers to other docs, which can point to other docs, etc.
- This creates a «web» of resources, called the **World Wide Web** (WWW)

Related Standards and RFCs

- Related RFCs
  - RFC 1945: Hypertext Transfer Protocol, HTTP/1.0 (in 1996)
  - RFC 2616: Hypertext Transfer Protocol, HTTP/1.1 (in 1999)
  - RFC 1738: Uniform Resource Locators (URL)
  - RFC 3986: Uniform Resource Identifier (URI): Generic Syntax (including URLs)
  - HTTP/2.0 http://tools.ietf.org/html/draft-ietf-httpbis-http2

- Standardization of Web-related mechanisms (HTML, webRTC)
  - World Wide Web Consortium (W3C): http://www.w3.org/

About the WWW

- Historic Background
- Communication in the WWW (HTTP, WebSocket, RTCWeb)
- HTTP Cookies
- Proxies
- Scaling Web Server Architecture

Towards the WWW

Brief history towards the WWW

1945

- Vannever Bush sketched the hypertext concept
  - It was called „Memex"

1962

- Doug Engelbart started work on implementation of hypertext model
  - Everything run on one computer

1963

- Ted Nelson coined the term „hypertext

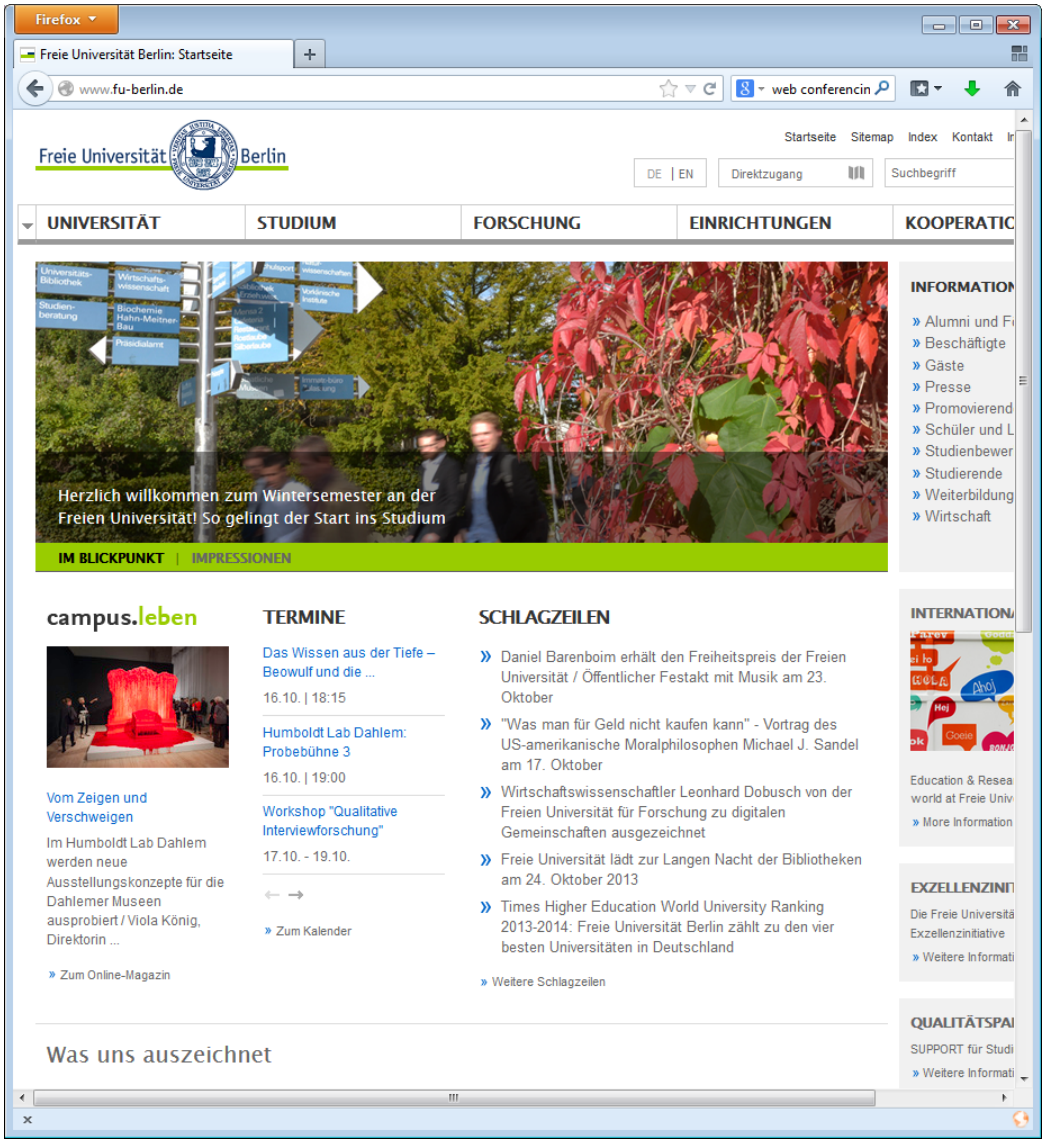
1989

- Tim Berners Lee introduced WWW
  - Connecting hypertext idea to TCP/IP and DNS

- World Wide Web (WWW) today gives access to interlinked documents distributed over several computers in the Internet
  - It uses the Internet as a data delivery service
  - It was developed to exchange data, figures, etc. between a large number of geographically distributed project partners via Internet

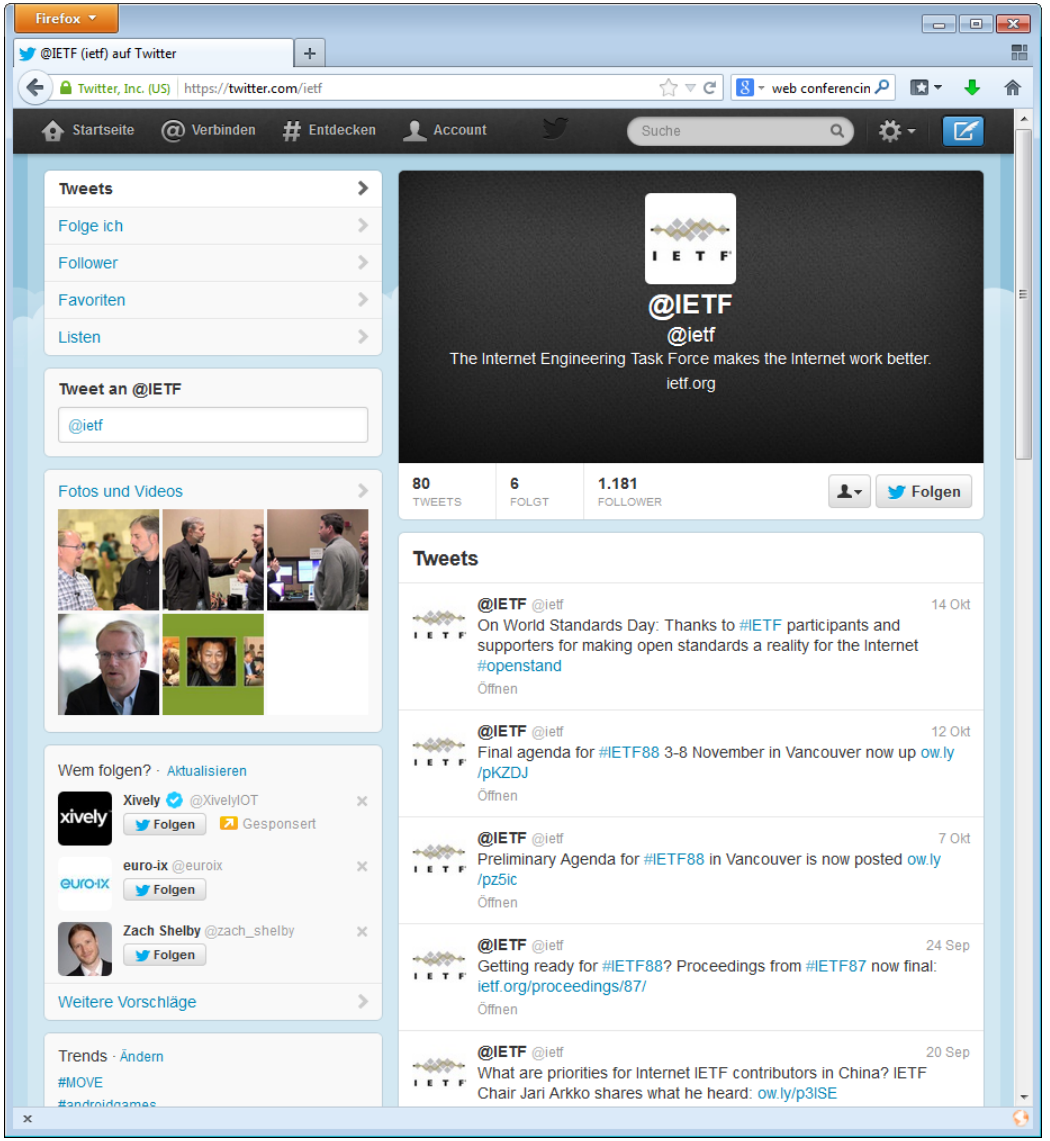
Evolution of the WWW: Serving Users

- We started we static web pages ...



Evolution of the WWW: Serving Prosumers

- … continued with interactive content …



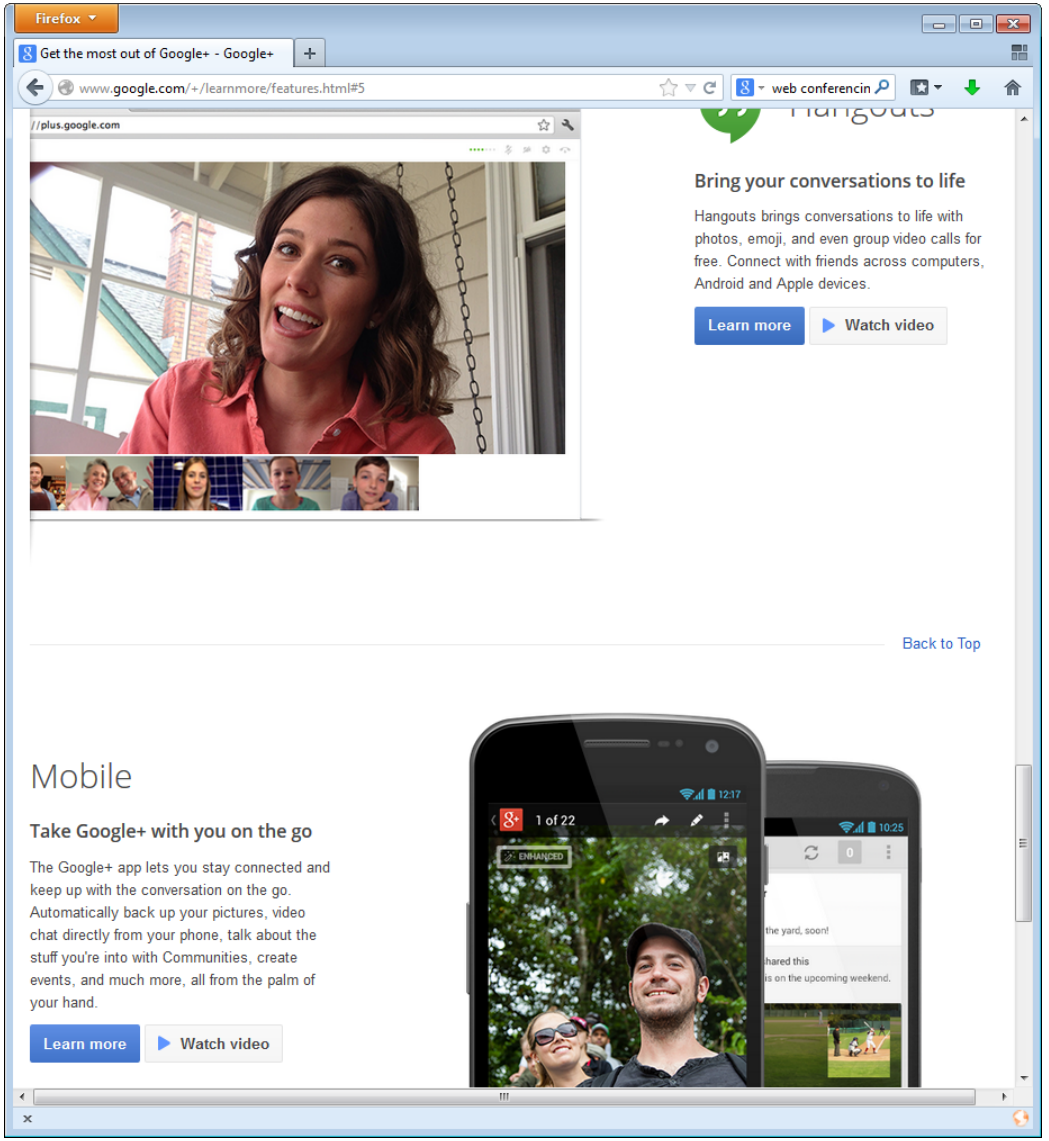
Evolution of the WWW: towards Real-Time

- ... currently focus on real-time communication ...

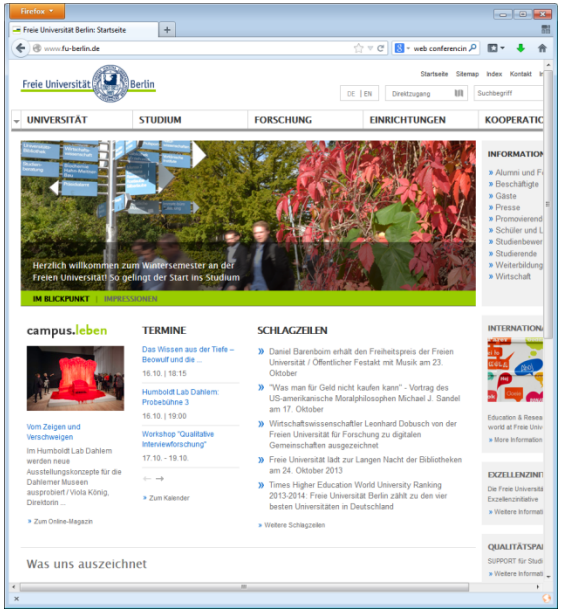
How do you bring content to the end user?

Let's start thinking about communication protocols!

Which properties have the different applications?

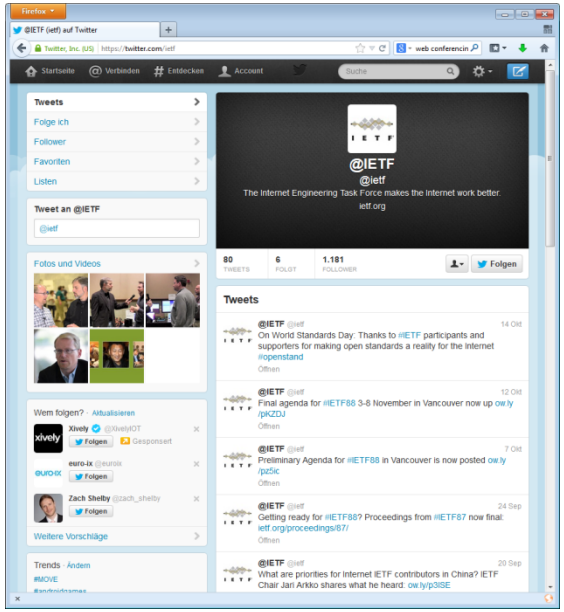


Requirements of Different WWW Applications



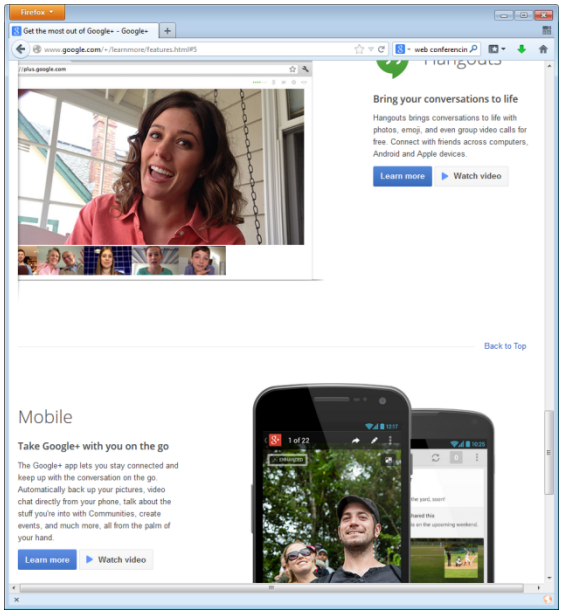
- Handling a huge user base

⇒Client/server model fits (pull paradigm)



- Changing content dynamically

⇒ Push paradigm preferred



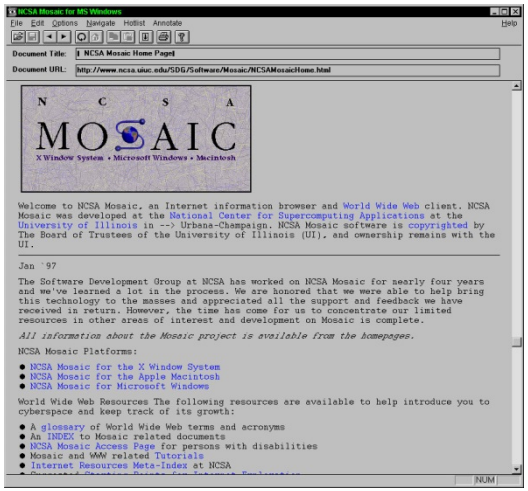
- Low latency, towards real time

⇒Direct communication between end users preferred

About the WWW

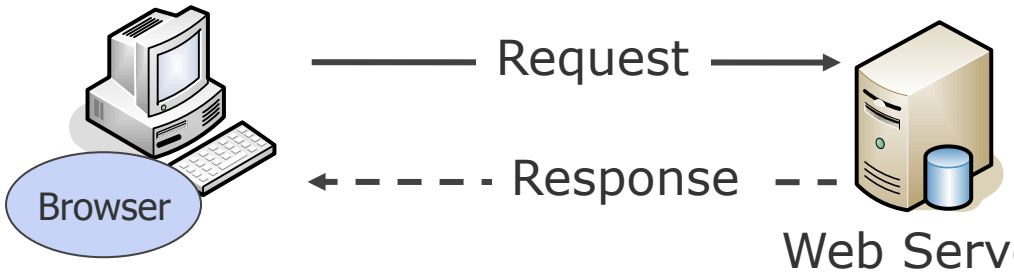
- Historic Background
- Communication in the WWW (HTTP, WebSocket, RTCWeb)
- HTTP Cookies
- Proxies
- Scaling Web Server Architecture

Standard Communication in the WWW



First graphic browser version (1993). Credit:National Center for Supercomputing Applications/University of Illinois Board of Trustees.

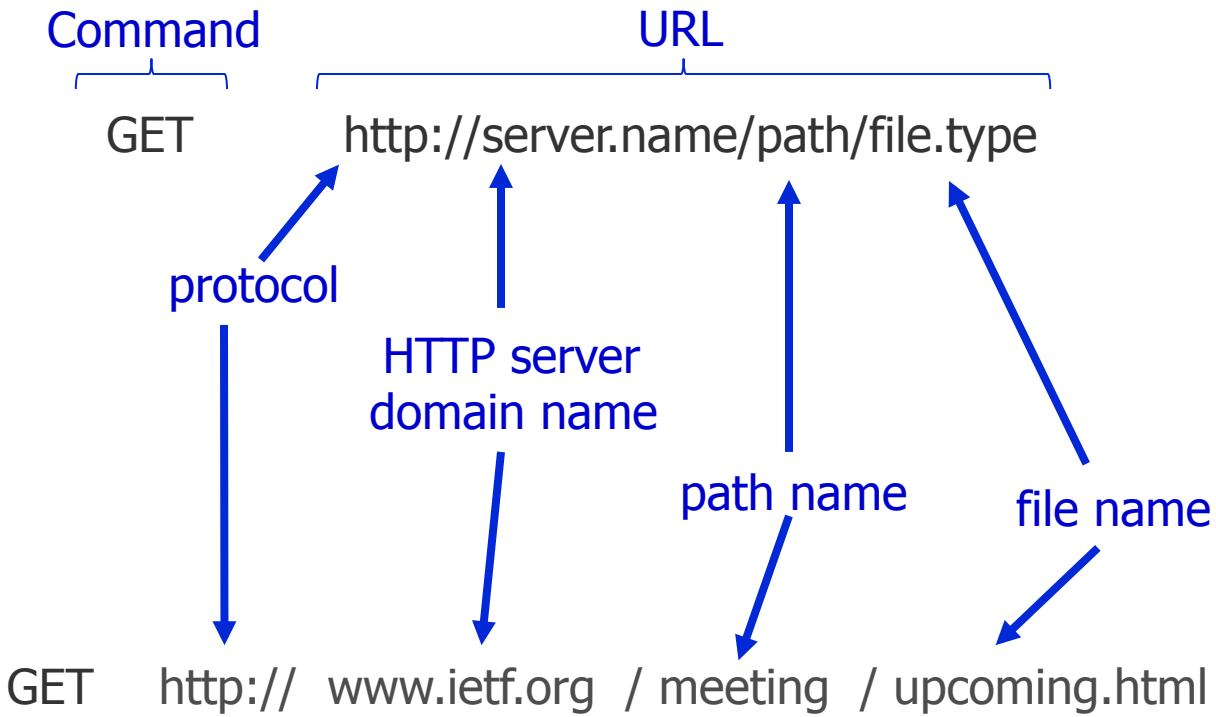
- **Client** = Browser
  - Popular browsers: Firefox, Internet Explorer, Chrome, Safari, Opera
- **Server** = Web server
  - Popular implementations: Ap Microsoft Internet Informatio Server, Tornado



HTTP – HyperText Transfer Protocol

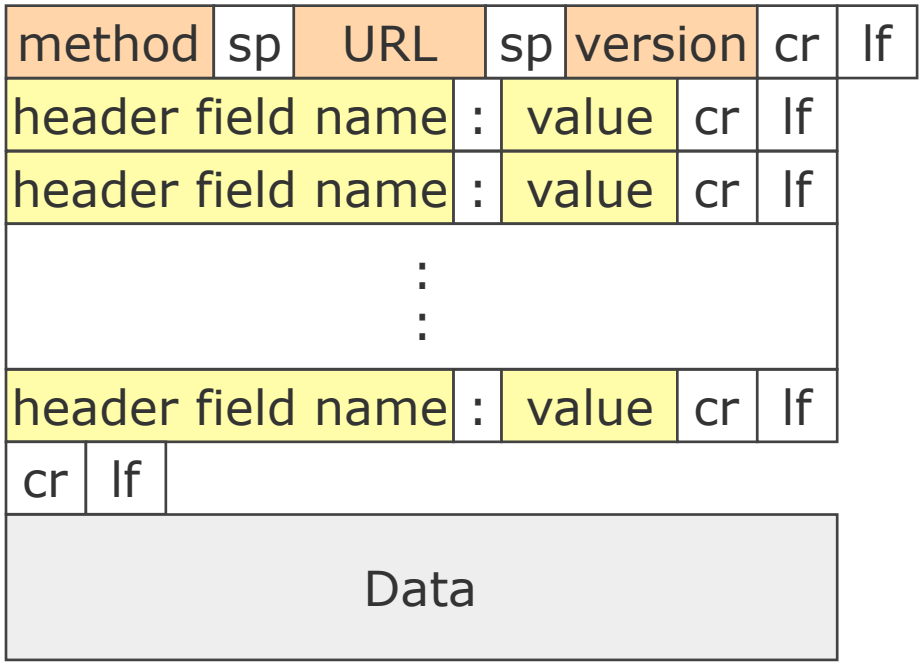
- Standard protocol used between a web server and a web client
  - If you load a web page, you use HTTP
- Server usually listens on well-known port 80
- Is request/response protocol
  - Client requests data, server responses
- Is an ASCII protocol (text-based)
- Identifies requested data by URL
- Is "stateless"
  - Server maintains no information about past client requests
  - Protocols that are "stateful" are complex!
    - Past history (state) must be maintained
    - If server/client crashes, their views of "state" may be inconsistent, must be reconciled
- HTTP provides an API to the web that complies with REST architecture

HTTP Requests



- Specified commands are:
  - GET: Load a web page
  - HEAD: Load only the header of a web page (not the content). For debugging.
  - PUT: Store a web page on the server
  - POST: Append something to the request passed to the web server
  - DELETE: Delete a web page

HTTP Request Format



Request line: necessary part, e.g.,

GET server.name/path/file.type

Header lines: optionally, further information to the host/document, e.g.

Host: www.fu-berlin.de

Accept-language: fr

User-agent: Opera /5.0

Entity Body: optionally. Further data, if the Client transmits data (POST method)

sp: space
cr/lf: carriage return/line feed

HTTP Response

status line
(protocol status code status phrase)

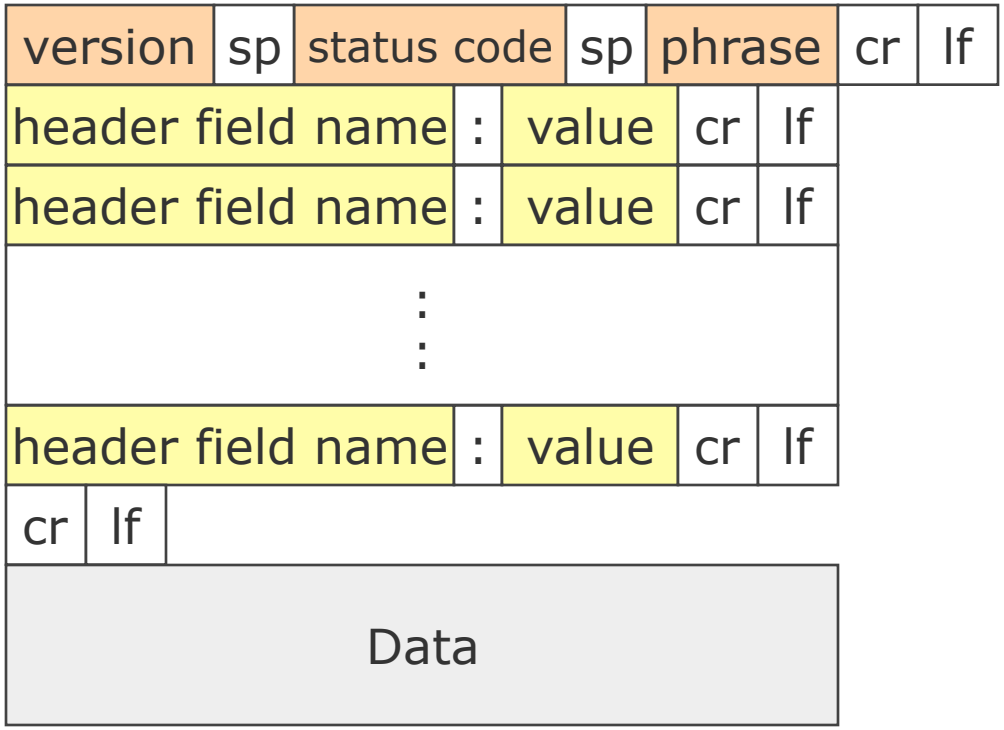
header lines

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html
```

data, e.g., requested HTML file

```
data data data data data ...
```

HTTP Response Format



Entity Body: inquired data

Status LINE: status code and phrase indicate the result of an inquiry and an associated message, e.g.

200 OK

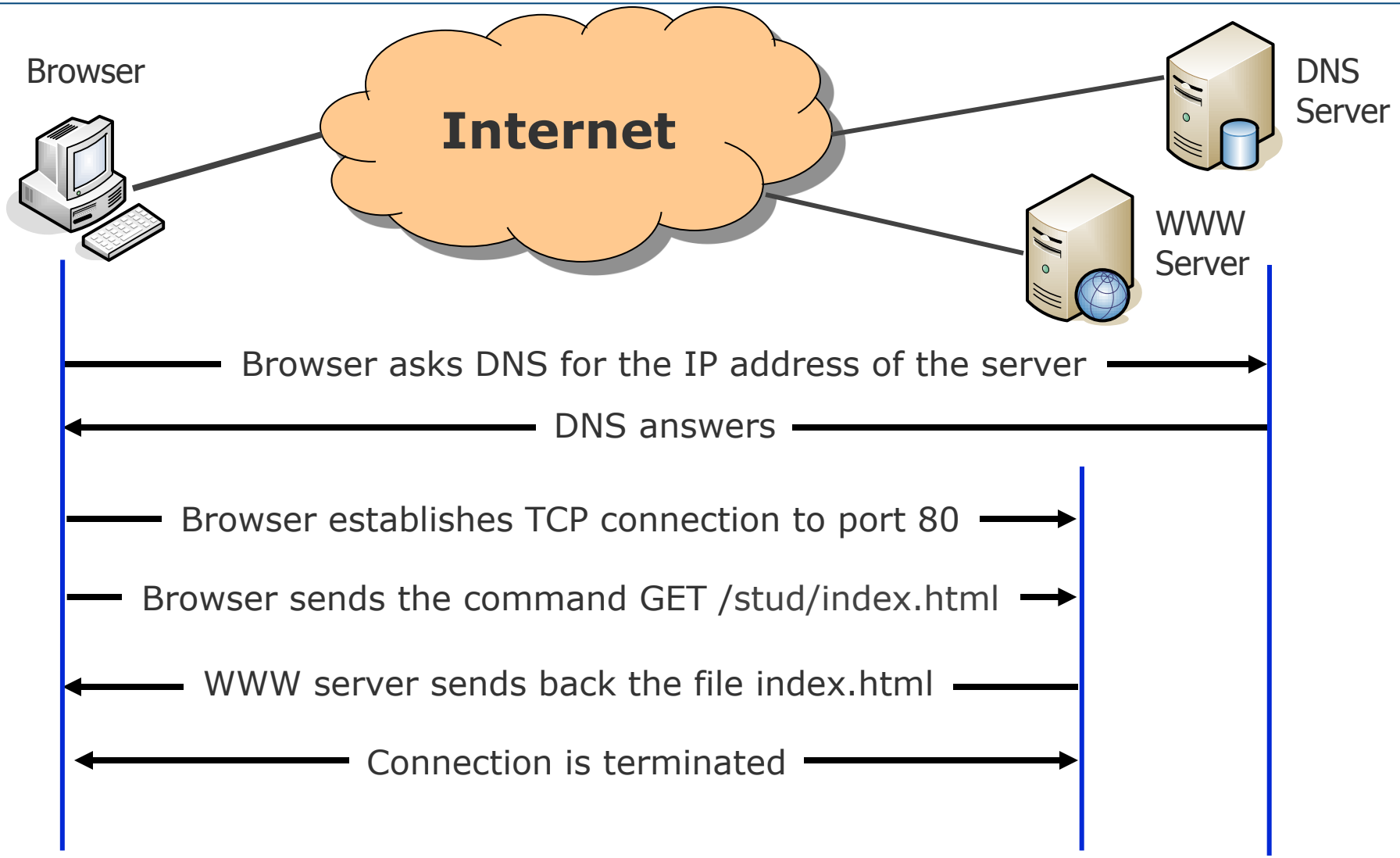
404 Not Found

400 Bad Request

Groups of status messages:

1xx: Only for information
2xx: Successful inquiry
3xx: Further activities are necessary
4xx: Client error (syntax)
5xx: Server error

Loading of Web Pages



Open Problems

1. How does a web server push content to the client?

- HTTP is a request/response protocol, client needs to poll explicitly
  - Large overhead

⇒ Solution: WebSocket Protocol (RFC 6455)

2. How do you enable communication between browsers?

- HTTP is a client/server protocol
  - Do you really want to integrate a web server into your web browser?

⇒ Solution: RTCWeb (under standardization, see IETF rtcweb group)

New Communication in the WWW: WebSockets

Overview

- Allows efficient, full-duplex communication between web server and client
- It does not need HTTP
- Web socket URLS, example: ws://www.example.com/newsfeed

Parallel Use of WebSocket Protocol and HTTP

- HTTP and WebSocket share port 80
- Connection request by client looks like HTTP GET request with Upgrade offer (upgrading HTTP connection into TCP socket)
- HTTP server replies with „Switching Protocols"

Available implementations

- Clients: Chrome, Firefox, Internet Explorer, Safari, Opera
- Server: Apache …

Future Communication in the WWW: WebRTC

Overview

- Implements real-time communication between web browsers
  - Communication does *not* require web server
  - Brings P2P architecture support to browsers
- Joint standardization activity between IETF (rtcweb) and W3C (WebRTC)
  - IETF defines protocol, W3C defines Javascript API
  - Standardization is ongoing
- Provides functions to establish media and data channel

Implementations

- Chrome, Firefox
- More expected ;)

About the WWW

- Historic Background
- Communication in the WWW (HTTP, WebSocket, RTCWeb)
- HTTP Cookies
- Proxies
- Scaling Web Server Architecture

From Stateless Protocol to Stateful Information

Problem with HTTP

- Stateless protocol
  - After an HTTP request/reply sequence. After each session, the web server "forgets" everything about this request/reply.
  - Advantages: simply scalable for large user base, little processing/memory
    - Enough for browsing
  - Drawbacks: cannot store "context", personalize user experience etc.
    - not suitable for applications like online shops which need to remember customers

Solution

- Cookie: tags assigned by the website, stored by the web browser
- Additional HTTP header field: Set-cookie
  - Instructs client to store the received cookie together with the server name
  - Client indicates cookie in its subsequent requests' header
  - The server is then able to identify related requests

⇒ Stateful information can be stored on server side

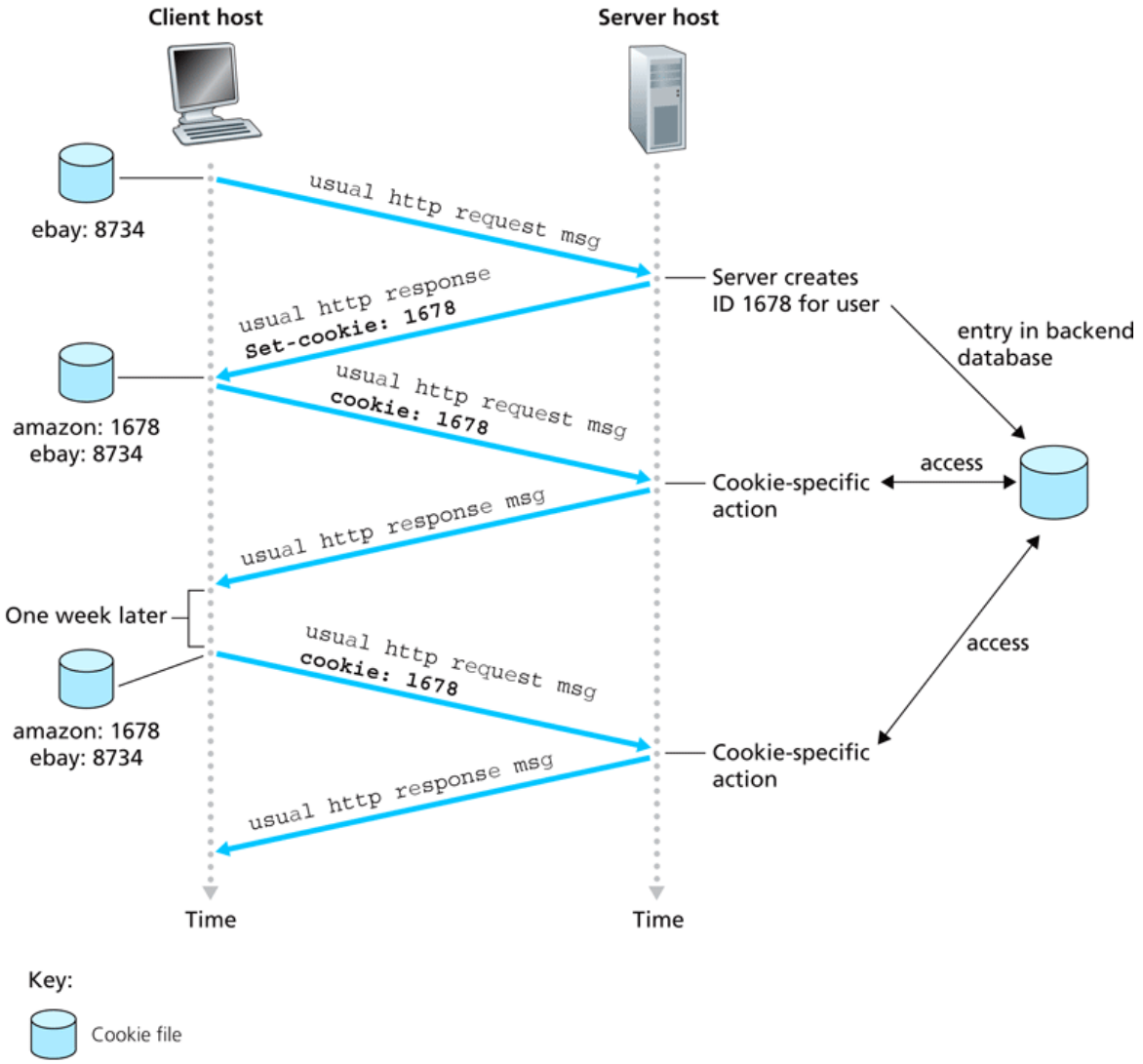
User Identification: Cookies

- Cookie database entries format:
  - Name-value pairs defined by the server for identification
  - Optional name-value pairs for, e.g., comments, date, TTL

Domain	Path	Content	Expires	Secure
toms-casino.com	/	CustomerID=497793521	15-10-08 17:00	Yes
joes-store.com	/	Cart=1-00501;1-07031	11-10-08 12:00	No
aportal.com	/	Prefs=Stk:SUNW+ORCL	31-12-08 17:30	No
sneaky.com	/	UserID=2344537333744	31-12-08 18:00	No

- A cookie is valid until its expiration time
- To delete a cookie: cookie resent with expiration time in the past
- Client sends a cookie together with next requests to the server

Cookie Exchange



Pros & Cons of Cookies

- What cookies enable:
  - Authorization
  - Shopping carts
  - Recommendations
  - User session state (Web email)
- Nice solution to manage "state":
  - Cookies: HTTP with persistent ID
  - Protocol maintaining state at sender/ receiver over multiple transactions
  - Simple management on the host side

Great. But what about privacy?

- Cookies can tell a LOT about you
  - Check your browser: hundreds of cookies? What is tracked for each cookie?
  - You can disable cookies, but many websites don't function anymore
- Global tracking with third-party cookies
  - Goal = gather info about ALL your online activity, to extract behavioral information
  - This info = really BIG business (e.g. deals between content providers and Doubleclick)
  - Disabled third party cookies? Tracking persists, with ad banners hosted by Doubleclick.
- There is cookie-less tracking too!
  - Advanced techniques using Javascript, caching, conditional HTTP requests and Etag
  - Check http://lucb1e.com/rp/cookielesscookies/

Another Good Way to Remember the Past

- **Nonpersistent HTTP**
  - At most one object is sent over a TCP single connection.
  - HTTP/1.0 uses nonpersistent connections
- **Persistent HTTP**
  - Multiple objects can be sent over single TCP connection between client and server
  - HTTP/1.1 uses persistent connections in default mode

Why switching from nonpersistent to persistent HTTP?

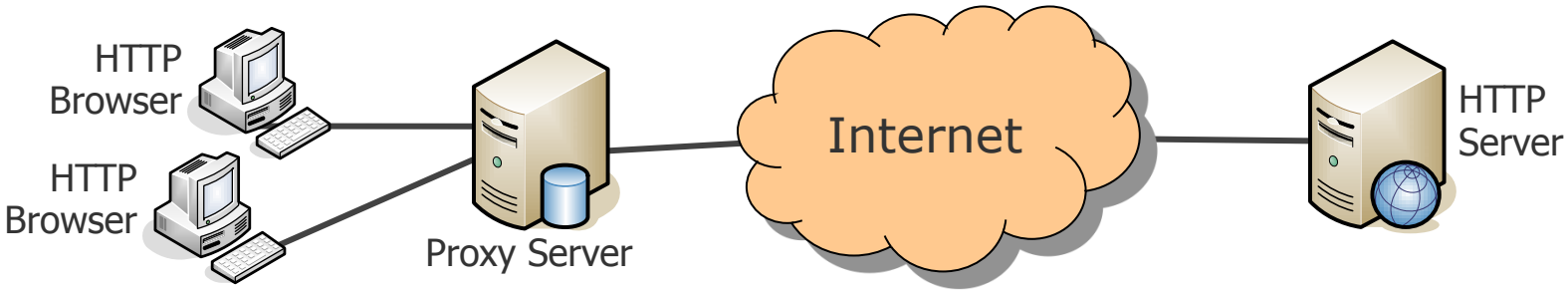
- TCP connection setup is expensive (CPU, memory, delay, etc.)
- Web page includes multiple objects (images …)
  - Using nonpersistent HTTP: One TCP connection per object request
- Persistent HTTP allows to send multiple HTTP request/responses over one TCP connection
- But: What happens if objects are distributed over multiple servers?

About the WWW

- Historic Background
- Communication in the WWW (HTTP, WebSocket, RTCWeb)
- HTTP Cookies
- Proxies
- Scaling Web Server Architecture

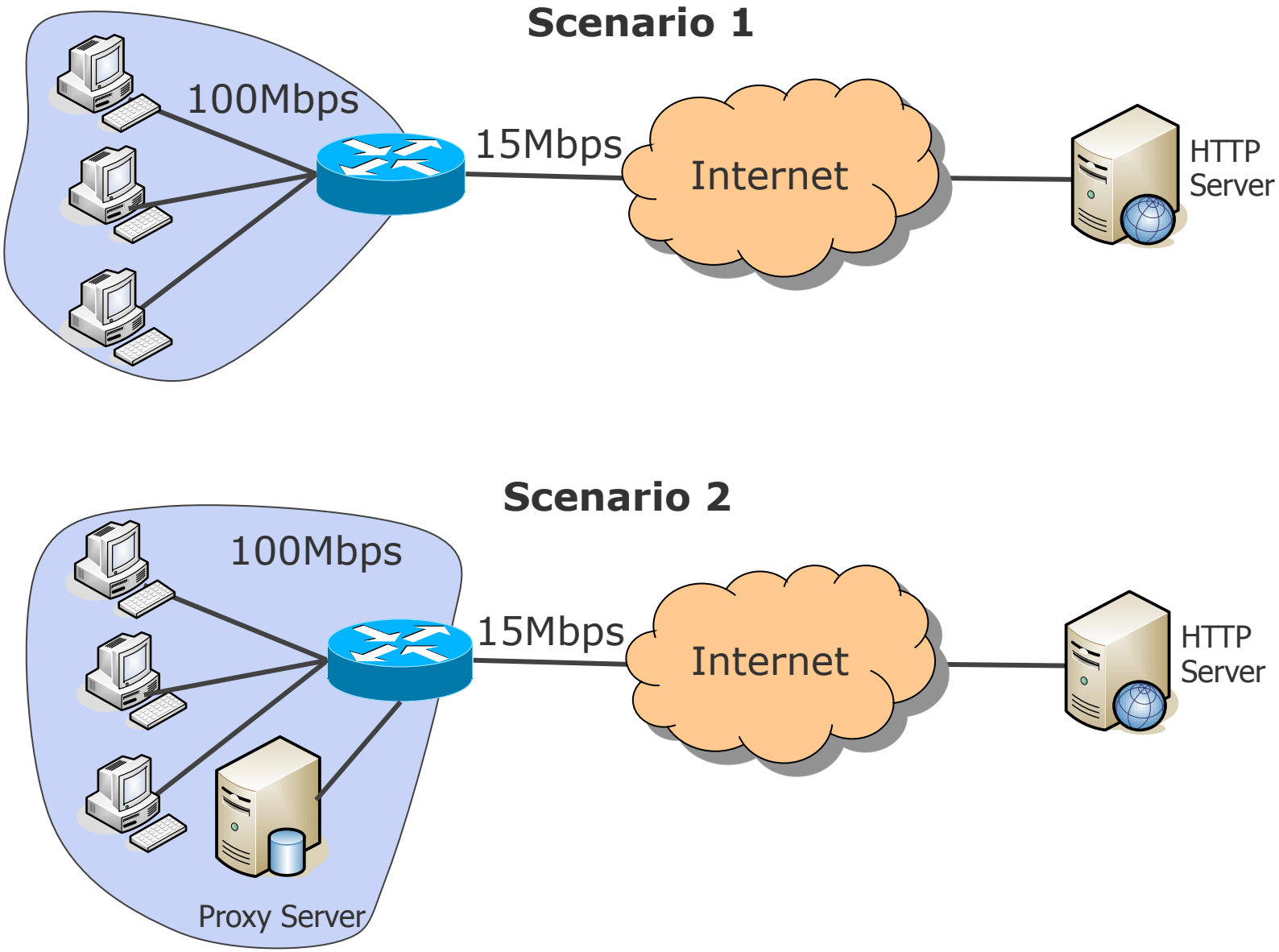
Proxy Server

- A Web Proxy is an intermediate entity used by several browsers. It takes over tasks of the browsers (complexity) and servers for more efficient page loading!



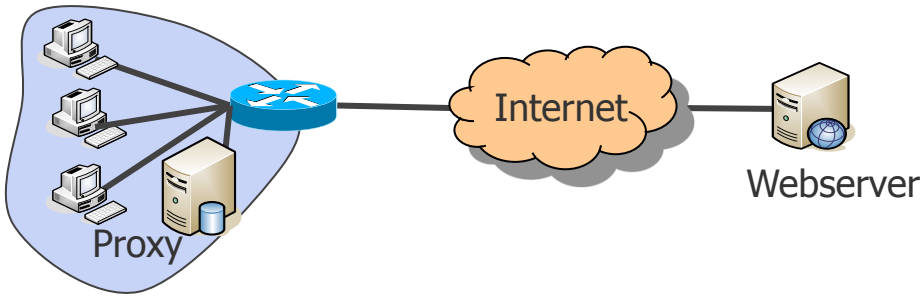
- Caching of web pages
  - A proxy temporarily stores the pages loaded by browsers. If a page is requested by a browser which already is in the cache, the proxy controls whether the page has changed since storing it. If not, the page can be passed back from the cache. If yes, the page is normally loaded from the server and again stored in the cache, replacing the old version.
- Support when using additional protocols
  - A browser enables also access to FTP, Telnet servers or even News, Gopher,
  - Instead of implementing all protocols in the browser, it can be realized by the proxy. The proxy then "speaks" HTTP with the browser and e.g. FTP with a FTP server.
- Integration into a Firewall
  - The proxy can deny the access to certain web pages (e.g. in schools).

Scalability: Delay & Throughput Gains with a Proxy



Scalability: Delay & Throughput Gains with a Proxy

- Scenario Characteristics
  - 100 Mbps Ethernet access link
  - Internet connection with 15 Mbps
  - Traffic characteristic
    - 15 requests per sec
    - Every request is for 1 Mbit of content (HTTP signaling thus negligible)
    - Internet delay ~2 sec
    - Access link delay ~0.01 sec
  - With proxy server
    - Hit rate of 0.4



- Performance considerations
  - Load on the access link

$$\frac{15\,\frac{\text{req}}{s}\times 1\,\text{Mbit}}{100\,\text{Mbps}} = 0.15$$

  - Load to the Internet without proxy

$$\frac{15\,\frac{\text{req}}{s}\times 1\,\text{Mbit}}{15\,\text{Mbps}} = 1.0$$

  - Load to the Internet with proxy

$$\frac{15\,\frac{\text{req}}{s}\times 1\,\text{Mbit}\times 0.6}{15\,\text{Mbps}} = 0.6$$

  - Delay without proxy ~ 2 sec
  - Delay with proxy
    0.4 x 0.01 s + 0.6 x 2 s ~ 1.2 s

33% less load to the Internet. 40% less delay!

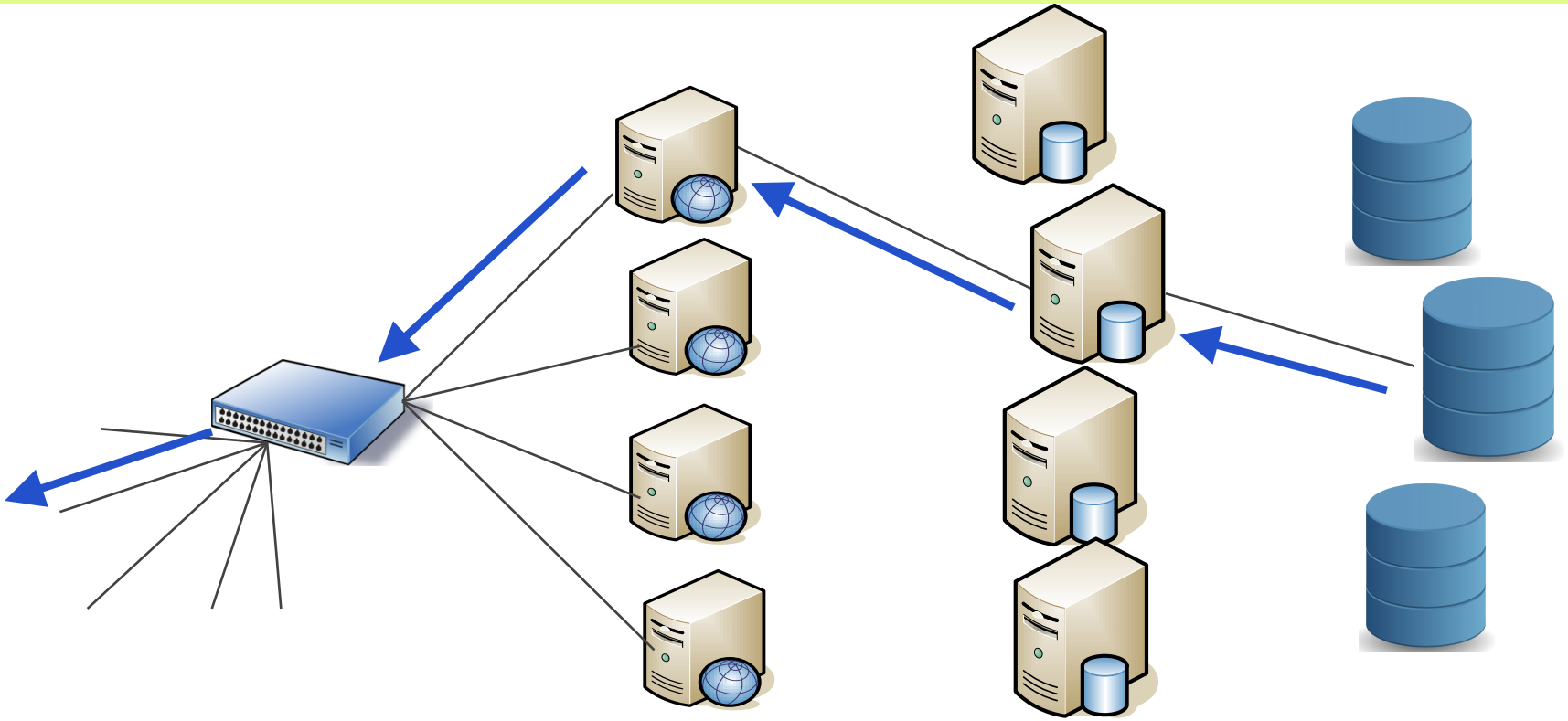
About the WWW

- Historic Background
- Communication in the WWW (HTTP, WebSocket, RTCWeb)
- HTTP Cookies
- Proxies
- Scaling Web Server Architecture

Big Web Servers: 3-tier Architecture

- Top 1000 websites: 90% of the overall web traffic. HUGE LOAD AHEAD!
- Need for specific scalable architecture

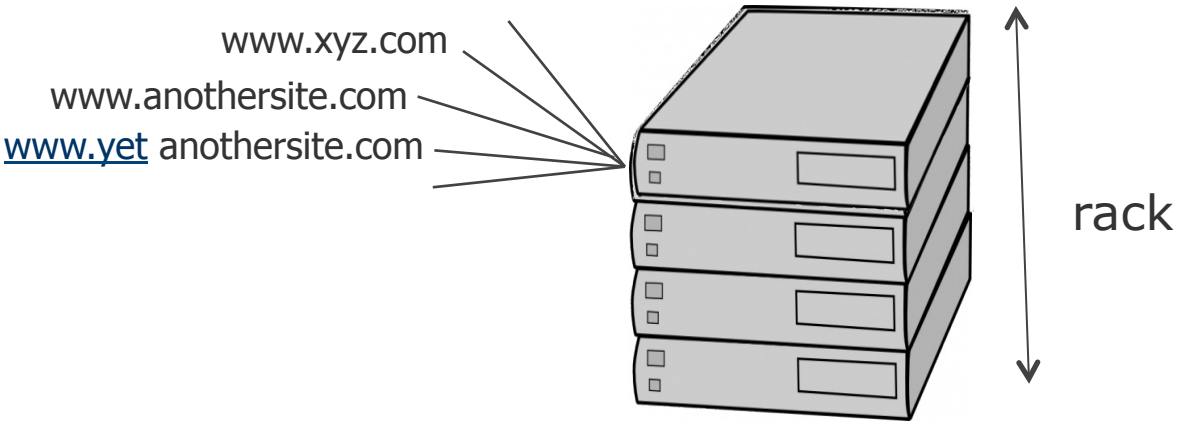
HTTP Requests	Load Balancer	Presentation Servers	Logic Servers	Database Storage



- 3-tier architecture can handle hundreds of thousands of hits/sec.

Medium & Small Webservers: Virtualization

- Medium & small webservers deal with more sporadic traffic
- Solution: pack thousands of «virtual» servers on a single machine



- Some companies focus on managing stacks of such machines (= data center)
- Multiplexing done with the server URL field in HTTP requests
- Each machine can handle approx. 100 hits per second