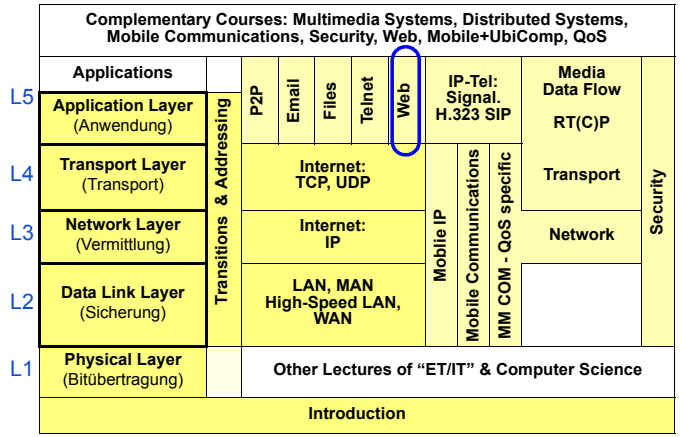
Communication Systems
Network Applications - Web

Prof. Dr.-Ing. **Lars Wolf**

TU Braunschweig
Institut für Betriebssysteme und Rechnerverbund

Mühlenpfordtstraße 23, 38106 Braunschweig, Germany
Email: wolf@ibr.cs.tu-bs.de

Scope



Overview

1. **The "Web": Introduction**
2. **WWW Architecture**
3. **Client - Server Communication: HTTP**
4. **Examples for HTTP Requests**
5. **HTTP: From Initial V. 1.0 to Current Versions**
6. **Document Structure**

Goal
- **overview with focus on communications**

Non goal
- **in detail, e.g., xml, web engineering, etc. (see related lectures)**

1. The "Web": Introduction

Original problem:
- **to present complicated experiments including diagrams and pictures**
  - research groups at different locations

Solution: World Wide Web (WWW, W3, „The Web"):
- **framework for hyperlink documents**
- **large collection of documents distributed all over the internet**
  - see also http://www.w3.org

History
- 1989 (March) Tim Berners-Lee (CERN, Geneva) publishes his first ideas
- 1993 (start) approx. 50 web-servers
- 1993 (Feb.) Mosaic distributes first version as shareware
- 1994 CERN and MIT found W3 Organization (W3O)
  Inria joins the developing W3 Consortium (W3C)
  objective: to promote the WWW
  (see also http://www.w3.org)
- 1995 (Nov.) html defined as HTML 2.0. in RFC 1866
  (ftp://ds.internic.net/rfc/rfc1866.txt)
- 1996 HTML 3.2 consensus for 1996
- since 1998 HTML 4.0 and very few variations
- semantic web to add knowledge to nodes (as metadata)

Other Background

Netscape (History & Background)

- 1952 Silicon Graphics (SGI) founded by Jim Clark
- 1993 Mark Andreesen develops Mosaic as a “front end” at the US National Center for Supercomputer Applications
- 1994 (April) J.Clark leaves Silicon Graphics
  Netscape Communic. founded by J.Clark & M.Andreesen
- 1996 Netscape Browser market share: approx. 70%
- 1998 available free of charge
- 1999 taken over by AOL
- today still an important player (but much less than in the past)

Sun Microsystems

- approx. 1994 Java as a Plug-in (Applet) defines additional functionality for browsers
- today the most important companies have Java under license

Microsoft

- since 1996 Internet Explorer (as part of the operating system)
- today major share of browser market (endusers)

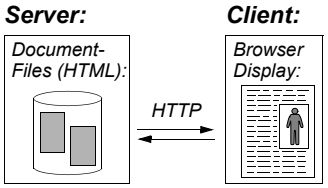
Mozilla

- since 1998 open source project (originally based on Netscape code)
- today code used by / feed back into Netscape

2. WWW Architecture

Paradigm: client-server architecture

- **server**
  - stores documents
- **clients (using browser)**
  - access documents
  - display them
  - integrate various media



Communication is done via a specific protocol

- **"Hypertext Transfer Protocol“ HTTP**
- **HTTP uses TCP/IP**

Documents

- **are written in „Hypertext Markup Language“ HTML**
- **are specified via Uniform Resource Locator (URL)**

Web Browser

Client uses browser to:

- **communicate with the server**
- **display documents**

Steps to display a document:

1. **determine URL**
2. **establish TCP connection to server**
3. **send request to retrieve document from server**
4. **interpret the contents, potentially requesting referenced files**
5. **generate local layout**
6. **display "layout"**

Most prevalent browsers:

- Microsoft Internet Explorer
- Netscape Navigator
- Opera
- Mozilla
- Mosaic
- Lynx (based on text)
- ...

Web Server

- **is contacted by client**
- **provides information back to client**

Basic steps (performed in loop):

- **accepts TCP connection from client**
- **gets name of file requested**
- **retrieves the file**
- **sends file as reply to the client**
- **releases TCP connection**

More features in modern web servers, e.g.:

- **caching**
- **multi-threaded, multi-processor, multi-tier, server-farm, ...**
- **generating data to be returned (from database, ...)**

Uniform Resource Locator (URL)

URL is the „address“of a page

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

- http://<host>:<port>/<path>?<searchpart>
- ftp://<user>:<password>@<host>:<cwd1>..<cwdN>/<name>;type=<typecode>
- mailto:<rfc822-addr-spec>
- nntp://<host>:<port>/<newsgroup-name>/<article-number>
- telnet://<user>:<password>@<host>:<port>
- file://<host>/path

Typical URL consists of three parts:

- **protocol for accessing the page (http, ftp, mailto, ...)**
- **the name of the host administrating the page**
- **the local name of the page on the host**

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

3. Client - Server Communication: HTTP

Communication sequence:

- **client**
  - connects to the server using TCP
  - usually uses Port 80
  - client places a request
- **server**
  - accepts TCP connection from client
  - gets name of file requested and retrieves the file
  - sends file as reply to the client
- **the TCP connection is closed (by server)**

HTTP - the document transfer protocol

- **„HyperText Transfer Protocol“**
- **defines permissible requests and replies**
- **request:**
  - simple ASCII message
  - (command plus parameters)
- **reply:**
  - document (and any data) within a MIME message format
    - MIME = Multipurpose Internet Mail Extensions

Client - Server Communication: HTTP (2)

HTTP / 1.0: (RFC 1945)

- **used by:**
  - CERN, NCSA, APACHE server
- **permits hypermedia access to resources**
  - provided by various applications
  - including those supported by SMTP, NNTP, FTP, Gopher, WAIS
- **task(s)**
  - to access and transfer multimedia contents
  - to transfer messages in a MIME-like format

Communication scheme:

- **open, operation, close**
- **request**
- **response**

Stateless:

- **each request is processed individually**
- **TCP connection is setup and released after request has been processed**
- **connections are of short duration only**

3.1 HTTP Request

Full request:

- **request line:**

```
Method SP(space)  Request-URL SP HTTP-Version CRLF
```

Example

```
GET  http://www.w3.org HTTP/1.0
```

- **plus**
  - general header (date, MIME version)

  and/or
  - request header (authorization, from, ..)

  and/or
  - entity header (allow, content type, expires,...)
  - CRLF
  - entity body

HTTP Requests

Each request begins with a method that has to be executed

Method	Description
GET	Request to read a web page
HEAD	Request to read the header of a web page
PUT	Request to store a web page on the server
POST	Attach data to a resource (e.g. news or forms)
DELETE	Delete a web page
LINK	Connect two existing resources
UNLINK	Cancel a connection between two resources

Method

- **in HTTP v1.0**
  - GET, HEAD and POST are the ones mainly used

Parameters

- **optionally, request header fields can be inserted in the lines following each respective parameter**

3.2 HTTP Response

Full response:

- **status line:**

  `HTTP version SP Status-Code SP Reason-Parameter CRLF`

  Example

  `... 200 OK`

- **plus**
  - general header (date, MIME version)

  and/or
  - response header (location, server, WWW authentications)

  and/or
  - entity header (allow, content type, expires,...)
  - CRLF
  - entity body

HTTP Response: HTTP v1.0 Status Codes

1xx: **Reserved for future use.**

2xx: **Success.**

- **200:** **OK.**
- **201:** **Created.**
- **202:** **Accepted.**
- **204:** **No Content.**

3xx: **Redirection**

- **301:** **Permanently moved to a different location.**
- **302:** **Temporarily moved to a different location.**
- **304:** **Not modified.**

4xx: **Client error.**

- **400:** **Wrong syntax.**
- **401:** **Unauthorized access.**
- **403:** **Forbidden access.**
- **404:** **Document not found.**

5xx: **Server error.**

- **500:** **Internal server error.**
- **501:** **Function not implemented.**
- **502:** **Bad Gateway.**
- **503:** **Service not available (temporarily).**

4. Examples for HTTP Requests

Example university

$ TELNET WWW 80

```
Trying 134.169.34.18...
Connected to agitator.ibr.cs.tu-bs.de.
Escape character is '^]'.
```

GET /INDEX.HTML HTTP/1.0

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<!-- Generated by strauss@ibr.cs.tu-bs.de at 2003-01-
29 17:32 -->
<html>
<head>
<meta content="text/html; charset=iso-8859-1" http-
equiv="Content-Type">
<title>Institut für Betriebssysteme und
Rechnerverbund</title>
...
```

Examples for HTTP Requests (2)

Example w3.org (with syntax error, where?)

```
$ TELNET WWW.W3.ORG 80
   Trying 18.23.0.23...
   Connected to www.w3.org.
   Escape character is '^]'.
GET HTTP://WWW.W3.ORG HTTP/1.0
.. BLANK LINE WITH <CRLF>
   HTTP/1.1 302 Moved Temporarily
   Date: Sat, 24 Jan 1998 12:43:10 GMT
   Server: Apache/1.2.5
   Location: http://www.w3.org/WWW
   Connection: close
   Content-Type: text/html
   <HTML><HEAD><TITLE>302 Moved Temporarily</TITLE></
   HEAD><BODY>
   <H1>Moved Temporarily</H1>
   The document has moved <A HREF="http://www.w3.org/
   WWW">here</A>.<P>
   </BODY></HTML>
   Connection closed by foreign host.
```


Examples for HTTP Requests (3)

```
$ TELNET WWW.W3.ORG 80
   Trying 18.23.0.23...
   Connected to www.w3.org.
   Escape character is '^]'.

GET HTTP://WWW.W3.ORG/ HTTP/1.0
.. BLANK LINE WITH <CRLF>
   Server: Apache/1.2.5
   Last-Modified: Sat, 09 Aug 1997 17:25:46 GMT
   ETag: "2d1d66-3ab-33eca81a"
   Content-Length: 939
   Accept-Ranges: bytes
   Connection: close
   Content-Type: text/html; charset=ISO-8859-1
   <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
   <HTML>
    <HEAD>
    <TITLE>
   .. and so on
```


Examples for HTTP Requests (4)

```
   <HTML>
    <HEAD>
    <TITLE>
   .. and so on
   <P class=policyfooter>
   <SMALL><A href="./Consortium/Legal/ipr-
   notice.html#Copyright">Copyright</A>
    &copy;  1997 <A href="http://www.w3.org">W3C</A>
   (<A href="http://www.lcs.mit.edu">MIT</A>,
   <A href="http://www.inria.fr/">INRIA</A>,
   <A href="http://www.keio.ac.jp/">Keio</A> ), All Rights Reserved.
   W3C
   <A href="./Consortium/Legal/ipr-notice.html#Legal
   Disclaimer">liability,</A>
   <A href="./Consortium/Legal/ipr-notice.html#W3C
   Trademarks">trademark</A>,
   <A href="./Consortium/Legal/copyright-documents.html">document use
   </A>and
   <A href="./Consortium/Legal/copyright-software.html">software
   licensing
   </A>rules apply. Your interactions with this site are in accordance
   with
   our <A href="./Consortium/Legal/privacy-
   statement.html#Public">public</A>
   and <A href="./Consortium/Legal/privacy-
   statement.html#Members">Member</A>
   privacy statements.</SMALL>
   </BODY></HTML>
   Connection closed by foreign host.
```


5. HTTP: From Initial V. 1.0 to Current Versions

Problems in HTTP / 1.0

- **limited to only ONE URL per TCP connection**
- **disconnect**
  - causes loss of any congestion control
  - may congest low bandwidth links
    - problems with flow control during connect and disconnect in TCP
- **server administrates a large amount of connections in close-wait state**
- **HTTP 1.0 uses**
  - more time for waiting
  - than for actual data transfer

HTTP characteristics / 1.1 (RFC 2086) and follow-on

- **implemented in JIGSAW, APACHE 1.2b, ...**
- **persistent connection**
- **cache characteristic**
- **new request methods**
- **range request**

HTTP v1.1: Methods

Method	Description
OPTIONS	Inquires about available communication options.
GET	Request to read a web page.
HEAD	Request to read the headers of a web page.
PUT	Request to store a web page on the server.
POST	Attach data to a resource (e.g. news).
PATCH	Like PUT, transferring varieties.
COPY	Copies a resource to a different location.
MOVE	Moves a resource to a different location.
DELETE	Deletes a web page.
LINK	Connects two existing resources.
UNLINK	Closes connection between two resources.
TRACE	Returns the request received from the server.
WRAPPED	Permits HTTP requests to be summarized.

HTTP Methods

HTTP permits an extendable amount of methods to display the purpose of a request:

- **GET:** **reads the data identified by the requested URL**
- **HEAD:** **reads any data header (containing information about data)**
- **PUT:** **stores any data at a URL**
- **POST:** **attaches data to a location specified by a URL**
- **DELETE:** **deletes data specified by a URL**
- **LINK:** **connects two resources**
- **UNLINK:** **closes existing connections**
- **.....**

RANGE:

- **requests one or more subranges of an entity**
  - instead of the complete entity

Persistent vs. Non-Persistent Connections

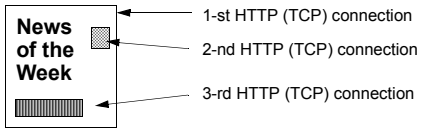
Performed steps in general:

1. **selection of object (clicking)**
2. **browser determines URL**
3. **DNS (Domain Name System) request to get IP address**
4. **browser establishes TCP connection to IP address / port 80**
5. **browser sends request (GET /...)**
6. **server returns requested file**
7. **closing TCP connection**
8. **browser displays content**
   - perhaps after interpretation of file and requesting of further files

Persistent vs. Non-Persistent Connections (2)

With non-persistent connection:

- **a separate TCP connection is established for every single URL requested**
  - TCP connection is closed after object is sent
  - hence, one request-response-pair per TCP connection
- **multiple parallel TCP connections possible (e.g., 5-10 with current browsers)**



Problems:

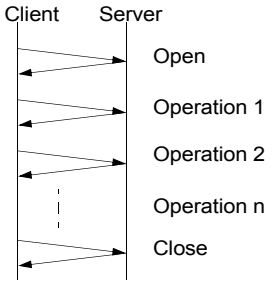
- **large resource demands in server**
- **slow-start, RTT determination, ...**

HTTP 1.0 provides non-persistent connections only

Persistent vs. Non-Persistent Connections (3)

Persistent connections:

- **establishing a single TCP connection**
  - to get multiple URLs from the same server
  - open, operations, close
- **are standard with each HTTP 1.1 connection**



Persistent connections have many benefits:

- **administrative overhead for TCP is reduced (CPU & memory)**
- **HTTP requests and responses can be sent on one connection representing a pipeline:**
  - pipelines permit the client to send several requests without waiting for responses
- **network congestion is reduced**
  - because the number of packets necessary to connect and disconnect is smaller

Caching in HTTP

The objective of caching in HTTP 1.1 is:

- **to reduce the amount of accesses to one and the same page, thereby**
  - avoiding repeated transmissions of the same data requests
    - reducing the access time (expiration mechanism)
  - avoiding repeated transmission of the same data, full responses
    - reducing the required net bandwidth (validation mechanism)

Cache control directives

- **restrictions with regard to**
  - what is supposed to be cached (server)
  - what is supposed to be stored in a cache (server / user agent)
- **modifications of expiration mechanism**
  - server / user agent
- **cache revalidation and reload control**
  - user agent

HTTP 1.1 Example

REQUEST

```
GET /index.html HTTP/1.1
Host: www.ibr.cs.tu-bs.de
```

RESPONSE

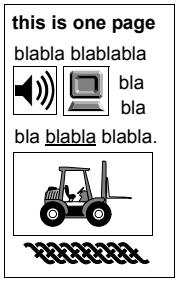
```
HTTP/1.1 200 OK
Date: Sat, 01 Feb 2003 14:41:32 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux PHP/4.2.3 mod_perl/1.26 mod_jk/1.1.0
Content-Location: index.html.de
Vary: negotiate,accept-language
TCN: choice
Last-Modified: Wed, 29 Jan 2003 16:32:39 GMT
ETag: "627-243d-3e380227;3e380228"
Accept-Ranges: bytes
Content-Length: 9277
Content-Type: text/html; charset=iso-8859-1
Content-Language: de

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Generated by strauss@ibr.cs.tu-bs.de at 2003-01-29 17:32 -->
<html>
<head>
<meta content="text/html; charset=iso-8859-1" http-equiv="Content-Type">
<title>Institut für Betriebssysteme und Rechnerverbund</title>
<meta name="author" content="">
<meta name="keywords" content="IBR, Betriebssysteme, Rechnerverbund">
<link rel="shortcut icon" href="/misc/shortcut_icon.gif">
<link rel="stylesheet" type="text/css" href="/ibr.css">
</head>
<body>
...
</body>
</html>
```


6. Document Structure

WWW documents („pages“) may consist of:

- **text**
- **icons**
- **drawings**
- **cards**
- **pictures**
- **audio clips**
- **video clips**



All media may contain links to other pages

Media may be displayed

- **directly via the browser itself or**
- **using an external „viewer“ (e.g. MPEG viewer)**

Documents: Internal Representation

Page presentation in HTML:

- **„Hypertext Markup Language“**
  - based on the SGML standard
  - defines “markup tags”
  - syntax and semantics
- **browser**
  - can interpret tags
  - can convert these into page layouts

HTML file:

```
<HTML>
<HEAD>My Page</HEAD>
<BODY>
This is my own Web page.
<P><B>Ain’t it nice?</B>
<P>Here’s my picture:
<IMG SRC=“myself.jpg“>
<P>That’s all for now!
</BODY>
</HTML>
```

important HTML tags:

- **<HEAD>...</HEAD>** — **page header**
- **<B>...</B>** — **text in bold print**
- **<P>** — **new paragraph**
- **<IMG SRC=“...“>** — **inserted image**
- **<A HREF=“...“>...</A>** — **link to another document**

Documents: Defining Hyperlinks

Tag <A> defines links:

- **format: <A HREF=“uniform resource locator“> item can be activated </A>**
- **example:**
  - HTML:

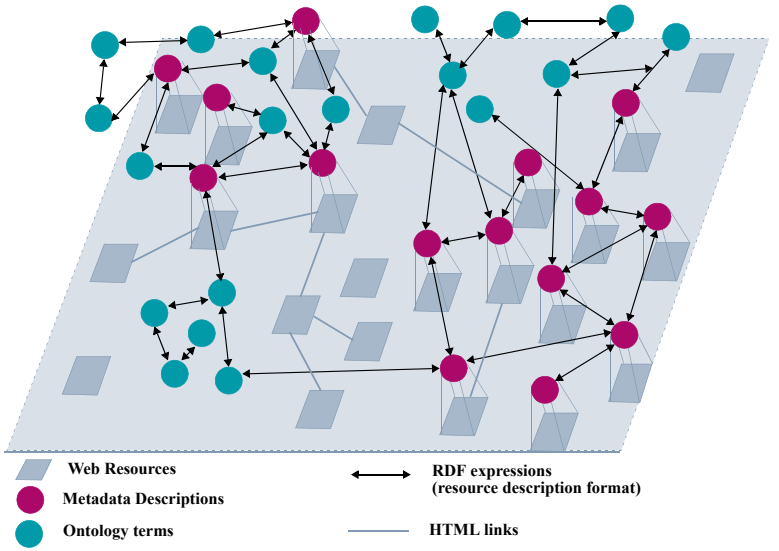
    „click <A HREF= “http://www.tu-braunschweig.de/index.html“> here</A> for TU Braunschweig“
  - layout:

    „click here for TU Braunschweig“
  - user entry:
    click here loads the document „www.tu-braunschweig.de/index.html“

HTML: Differences Between Various Previous Versions

	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0
Active maps and images		X	X	X
Equations			X	X
Forms		X	X	X
Hyperlinks	X	X	X	X
Images	X	X	X	X
Listen	X	X	X	X
Toolbars			X	X
Tables			X	X
Objects (Generalization of the IMG tag)				X
Formula				X

7. Future Evolution: Semantic Web



Future Evolution: Semantic Web (4)

Notion

"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**

see e.g.
- **http://www.w3.org/2001/sw/**
- **http://www.semanticweb.org/**
- **http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html**