Communication with a Web Server

- HTTP is based on TCP – to experiment with the protocol telnet can be used.

> telnet 129.13.170.1 80[RETURN]
GET /index.html HTTP/1.0[RETURN][RETURN]

> telnet www.teco.edu 80[RETURN]
HEAD /index.html HTTP/1.0[RETURN][RETURN]

**HTTP Transaction**

```
user

fetch resource

Client (Browser)          Server

Request

Analyzing
the content

Response

fetch resource
```

Transaction 1

Transaction 2
HTTP/0.9

- GET method
- simple, lightweight, fast, easy to implement
- Transfer of text documents (preferably HTML)
- Not specified in a RFC

![Diagram of HTTP/0.9 sequence]

DNS host name resolution

- Query: www.cs.comp-c.de
- Server: 1.2.3.4

HTTP/1.0

- only an informational RFC, 1992-1996 [RFC1945]
- message Types
  - request (GET, HEAD, POST)
  - response
- header fields
  - variable number of fields
  - Syntax: <field_name> " : " <field_value>
  - Transfer of meta information on the request, response, and content
- response codes
  - status and error information
- media types
  - transfer of arbitrary resources, especially
    - graphics, images, audio, video
  - based on MIME (multipurpose internet mail extensions)
- basic mechanism for access control and authentication

![Diagram of HTTP/1.0 scenario]
Document transfer (HTTP) I

- **GET Method**
- **URL:** `http://www.teco.edu/lehre/webe/index.html`
- **procedure at the client side**
  - Identify host name from the URL `www.teco.edu`
  - Resolve the IP-Address `129.13.170.1`
  - Identify port number `80` (default)
  - Open Socket (TCP) to `129.13.170.1` Port `80`
  - Send method over the Socket `GET /lehre/webe/index.html HTTP/1.0`
  - Read from socket until the socket is closed by the server.
  - Result: header with status and the requested resource or an error message

Document transfer (HTTP) II

- **GET Method**
- **URL:** `http://www.teco.edu/lehre/webe/index.html`
- **procedure at the server side**
  - A process on the machine `129.13.170.1` waits on port `80` for a connection request
  - If there is a request a connection is established, then:
    - Read from socket to the first empty line
    - Analyze the given request (extract method and resource name)
    - Write status on the socket
    - Localize resource (e.g. File system), read the resource and write it to the socket
    - Close the socket

Document transfer (HTTP) III

- **Input:** URL `http://www.teco.edu:8080/
- **Typical page:** `Sorry Not HTTP-Methods supported!`

Tools I

- Program shows the request of the browser as web page: `http://www.teco.edu:8080/`
- **Typical page:**

  **Sorry Not HTTP-Methods supported!**

  Your Request:
  GET / HTTP/1.1
  Accept: */*
  Referer: `http://www.teco.edu/lehre/webe/beispiele.html`
  Accept-Language: en-us
  Accept-Encoding: gzip, deflate
  User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt)
  Host: `www.teco.edu:8080`
  Connection: Keep-Alive
  Cookie: SITESEVER=ID=fe340799f17c660e09e1f34c9dbf
Tools II

Program to build and send HTTP-Requests:

```
http://www.teco.edu/lehre/webe/beispiele/http.html
```

---Open TCP connection to www.teco.edu:80
---Request:
GET /lehre/webe/ HTTP/1.1
Host: www.teco.edu
Accept: */*
Connection: keep-alive
---End of Request
---Server Reply:-------------
HTTP/1.1 200 OK
Date: Thu, 28 Oct 1999 10:29:07 GMT
Server: Apache/1.2.1
Keep-Alive: timeout=10, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html
b76
<html>
...
```

HTTP/1.0 Example

Request

```
GET /index.html HTTP/1.0
Accept: image/gif, image/x-xbitmap, image/jpeg, */*
Accept-Language: de
Accept-Encoding: gzip
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT)
```

Response

```
HTTP/1.0 200 OK
Server: ServerName
Content-Type: text/html
Content-Length: 3213

<html>
<head>
<title>Oracle Corporation - Home</title>
...
</head>
<body bgcolor="#ffffff" link="#000000" vlink="#ff0000">
...</body>
</html>
```

Documents contain Resources I

reply of the servers

```
HTTP/1.0 200 OK
Content-Type: text/html
Content-Length: 3213
```

```
<html>
<head>
<title>Oracle Corporation - Home</title>
...
</head>
<body bgcolor="#ffffff" link="#000000" vlink="#ff0000">
...
</body>
</html>
```

Tools III

- programs to analyze the traffic in a network:
  - tcpdump (Unix/Linux),
  - etherpeak (Mac) [www.wildpackets.com],
  - Systems Management Server (Windows NT)
- works only in superuser mode
- abuse is illegal!
Documents contain Resources I

- reply of the servers
- HTTP/1.0 200 OK
- Content-Type: text/html
- Content-Length: 3213

```
<html>
<head>
<title>Oracle Corporation - Home</title>
</head>
<body bgcolor="#ffffff" link="#000000" vlink="#ff0000">

...<input name=q size=10 maxlength=800 value="">

...<a href="/html/dev_it.html">
<img src="/images/dev_it_off.gif" border=0> </a>

...</body>
</html>
```

Documents contain Resources II

- images
- background
- buttons
- music
- audio

Documents contain Resources III

- images
- background
- buttons
- music
- audio

Documents contain Resources IV

```
Client (Browser)               Server

URL                             Request
analyze
image1                             Response
image2                             Request
...                              loadresource
...                              Response
image n                             Request
...                              loadresource
```

```
```
HTTP/1.0, Performance Problems

- see further reading:
  Simon E Spero, July 1994,
  Analysis of HTTP Performance problems
  http://www.w3.org/Protocols/HTTP/1.0/HTTPPerformance.html

- In HTTP most time is spent waiting and not transferring data due to the slow start mechanism in TCP

HTTP/1.0, Basic Authentication I

- restrict access to selective resources
- include information about the user (names) in access log file
- basic authentication
  - simple username - password scheme
  - `<user>:<passwd>` Base64 coded
    - Base64: groups of 24 Bits are encoded in 4 6-Bit characters (highly compatible subset of US-ASCII), in RFC1521
  - No encryption!
- procedure:
  - client requests a resource
  - server answers with status code: `401 Unauthorized` and header `WWW-Authenticate`
  - client requests a resource with additional header
    - `Authorization: <user>:<passwd>` (Base64 coded)
  - server checks `<user>:<passwd>` with access restrictions
  - If `<user>:<passwd>` is valid user will return the resource

HTTP/1.0, Basic Authentication II

- restrict access to selective resources
- include information about the user (names) in access log file
- basic authentication
  - simple username - password scheme
  - `<user>:<passwd>` Base64 coded
    - Base64: groups of 24 Bits are encoded in 4 6-Bit characters (highly compatible subset of US-ASCII), in RFC1521
  - No encryption!
- procedure:
  - client requests a resource
  - server answers with status code: `401 Unauthorized` and header `WWW-Authenticate`
  - client requests a resource with additional header
    - `Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==`
  - server checks `<user>:<passwd>` with access restrictions
  - If `<user>:<passwd>` is valid user will return the resource

HTTP/1.0 Example User und Password I

- request
  - `GET /index.html HTTP/1.0`
  - `Accept: image/gif, image/x-xbitmap, image/jpeg, */*`
  - `Accept-Language: de`
  - `User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT)`

- response
  - `HTTP/1.0 401 Access Denied`
  - `Server: Apache/1.2.1`
  - `WWW-Authenticate: Basic realm="teco150pc.teco.edu"`
  - `Content-Length: 24`
  - `Content-Type: text/html`
  - `Error: Access is Denied`
HTTP/1.0 Example
User und Password II

- browser will prompt user to input user name and password
- if there is a password stored from a previous access the browser will use this (transparently for the user)

HTTP/1.0 Beispiele
User und Passwort III

- request
  - GET /index.html HTTP/1.0
  - Accept: image/gif, image/x-xbitmap, image/jpeg, */*
  - Accept-Language: de
  - User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT)
  - Authorization: Basic YWxicmVjaHQ6dGVzdA==

- response
  - HTTP/1.1 200 OK
  - Server: Apache/1.2.1
  - Content-Length: 2989
  - Content-Type: text/html
  - <HTML>
  - <TITLE> ...

Critical Points/Problems with HTTP/1.0

- no support for non-IP-based virtual hosts
  - It is not possible to run more than one web server on a machine with only one IP-address
- only one request per connection
  - low performance with TCP (e.g. slow start, RFC2001)
- very basic caching model
  - no protocol level support for proxies and gateways
- no partial transfer of resources
  - high data value due to complete retransmission
  - disconnection results in complete retransmission
- insecure and simple authentication
  - password not encrypted
HTTP/1.1 – Abstract RFC 2616

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers (z.B. RFC2324). A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification defines the protocol referred to as "HTTP/1.1", and is an update to RFC 2068.

HTTP/1.1 at a Glance I

- application protocol (ISO layer 7)
- For cooperatively used distributed hypermedia systems.
- properties:
  - generic
  - state-less
  - object-oriented
  - open
  - support for typing
  - support for negotiation of data formats and representation
  - independent of data transmitted
- HTTP is used in the World-Wide Web since 1990
- current specification is "HTTP/1.1" (RFC2616)

HTTP-URLs - Details

```plaintext
<http_URL> = "http://" <host> [":" <port>] [<abs_path>]
```

```plaintext
<http> ::= (H|h)(T|t)(T|t)(P|p) - Caution!, Implementation is browser dependent
```

```plaintext
<host> ::= <DNS-Name> | <IP-Adresse>
```

```plaintext
www.teco.edu
teco16a.teco.uni-karlsruhe.de
web.de
129.13.170.1
```

```plaintext
<port> ::= <digits>
```

```plaintext
80 (Standard), 1080, 8080, 3128
```

```plaintext
<abs_path> ::= "/"[<path ["?"<query>]["#"<fragment>]]
```

```plaintext
/, /index.html, /cgi//../index.html,
/urlaub/photo.jpg, /lehre/webe/unterlagen.html#v1,
/cgi-bin/print.pl?name=Maier&alter=26,
/\%7Ealbrecht/,
\%/\%7Ealbrecht/,
```

HTTP-URLs - Examples

```sql
<http_URL> = "http://" <host> [":" <port>] [<abs_path>]
```

```sql
http://www.teco.edu/
http://www.teco.edu:80/index.html
```

```sql
http://www.teco.edu/lehre/webe/unterlagen.html#schnitt3
http://teco16a.teco.uni-karlsruhe.de/projects/ftp/screen.gif
```

```sql
http://teco16a.teco.uni-karlsruhe.de/~albrecht/urlaub/photo1.jpg
http://teco16a.teco.uni-karlsruhe.de/%7Ealbrecht/urlaub/photo1.jpg
```

```sql
http://teco16a.teco.edu:8080/cgi-bin/printenv
http://teco16a.teco.edu/cgi-bin/addr.pl?name=Maier&alter=26
http://www.altavista.com/cgi-bin/query?pg=4&q=+algorithm+base64
```

```sql
http://www.teco.edu/~albrecht/cgi//../index.html
```
**URI Comparison**

- When comparing two URIs to decide if they match or not, a client **SHOULD** use a case-sensitive octet-by-octet comparison of the entire URIs, with these exceptions:
  - A port that is empty or not given is equivalent to the default port for that URI-reference;
  - Comparisons of host names **MUST** be case-insensitive;
  - Comparisons of scheme names **MUST** be case-insensitive;
  - An empty abs_path is equivalent to an abs_path of "/".

- Characters other than those in the "reserved" and "unsafe" sets (see RFC 2396) are equivalent to their "% HEX HEX" encoding.

- URL comparison is necessary for caching

**<port> Comparison**

- same port number
- In case there is no port number then <port> is equal to 80
- examples
  - http://www.teco.edu/ is equal
  - http://www.teco.edu:80/
  - http://www.teco.edu:8080/ is not equal
  - http://www.teco.edu:80/

**<host> Comparison**

- <host> is an IP-address
  - same IP → host is equivalent
- <host> is a DNS name
  - same DNS name (case insensitive) → same host
  - DNS resolution to the same IP-address does not imply that hosts are equal! (non-IP based virtual hosts)
- example
  - www.TecO.EDU = WWW.TECO.EDU = www.teco.edu
  - www.teco.edu (IP=129.13.170.1) is not the same as wearable.teco.edu (IP=129.13.170.1)

**<abs_path> Comparison**

- <path> must be handled case sensitive.
  - Caution! Some implementations are not case sensitive (e.g. DOS/Windows)
- empty <path> is equal /
- when / is requested the server will reply with a directory listing, a designated file (e.g. index.html, index.htm, default.htm, etc.), or an error message (e.g. "Directory browsing not allowed"). Therefore there is not guaranty that / is equal to a certain file name.
- the parts <query> and <fragment> are case sensitive
Comparison

- Escaped encoding (RFC2396)
  - \texttt{escaped} = "$" hex hex
    \[ \text{hex} = \text{digit} | \text{"A"} | \text{"B"} | \text{"C"} | \text{"D"} | \text{"E"} | \text{"F"} | \text{"a"} | \text{"b"} | \text{"c"} | \text{"d"} | \text{"e"} | \text{"f"} \]
  - examples
    - \texttt{http://www.teco.edu} is equal \texttt{http://www.teco.edu/}
    - \texttt{http://www.teco.edu} ist not necessarily equal to \texttt{http://www.teco.edu/index.html}
    - \texttt{http://www.teco.edu/~albrecht/} is equal \texttt{http://www.teco.edu/%7ealbrecht/}
    - \texttt{http://www.teco.edu/%7ealbrecht/} is equal \texttt{http://www.teco.edu/%7Ealbrecht/}

Structure of HTTP Messages

- \texttt{HTTP-message} = \texttt{Request | Response}
- \texttt{generic-message} = \texttt{start-line}
  - \texttt{message-header}
  - \texttt{CRLF [ message-body ]}
- \texttt{start-line} = \texttt{Request-Line | Status-Line}
- \texttt{message-header} = \texttt{field-name ":" [ field-value ] CRLF}

Request

- \texttt{Request} = \texttt{Request-Line}
  - \texttt{* (general-header | request-header | entity-header ) CRLF}
    - \texttt{[ message-body ]}
- \texttt{Request-Line} = \texttt{Method SP Request-URI SP HTTP-Version CRLF}
- \texttt{Method} = \texttt{"OPTIONS" | "GET" | "HEAD" | "POST" | "PUT" | "DELETE" | "TRACE" | extension-method}
Request Header Fields

request-header = Accept
| Accept-Charset | Accept-Encoding | Accept-Language
| Authorization
| From
| Host
| If-Modified-Since | If-Match | If-None-Match
| If-Range | If-Unmodified-Since
| Max-Forwards
| Proxy-Authorization
| Range
| Referer
| User-Agent