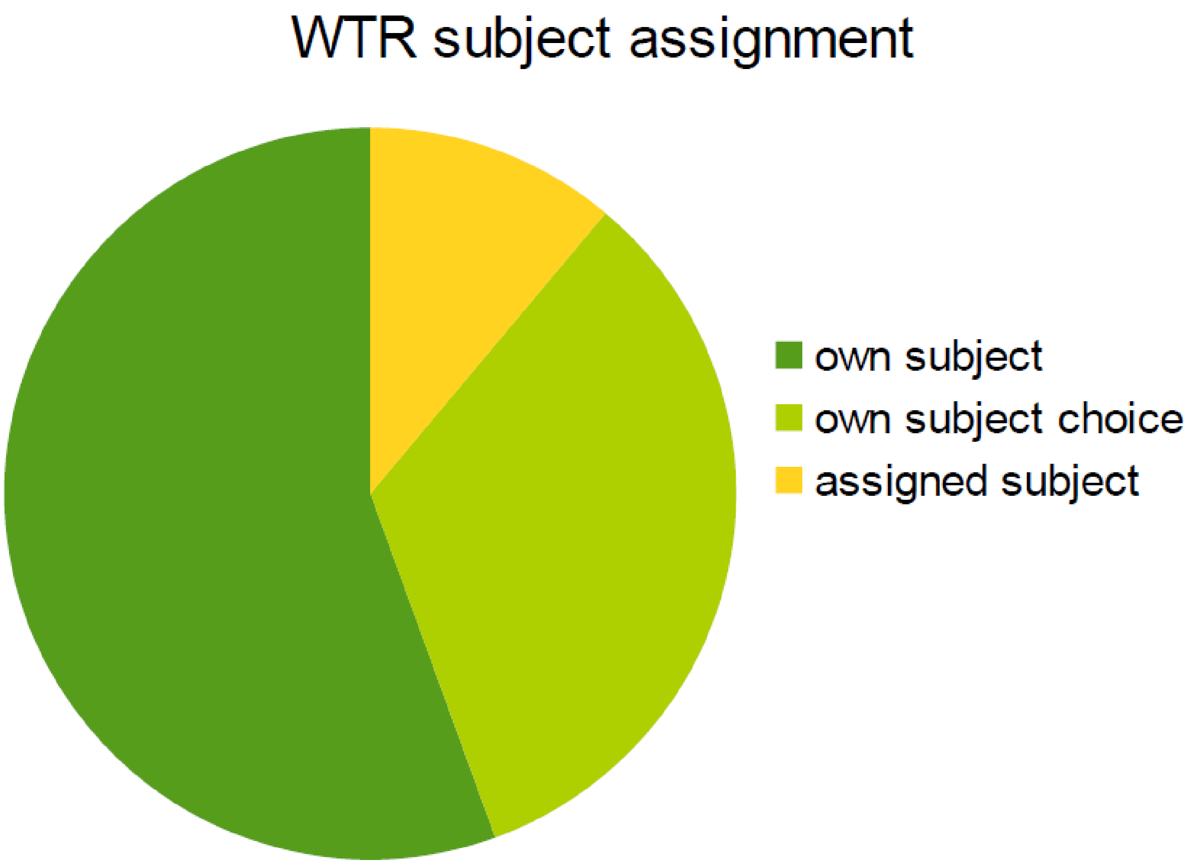
Web Technology 2015

Lecture 4. The World Wide Web: HTTP & HTML

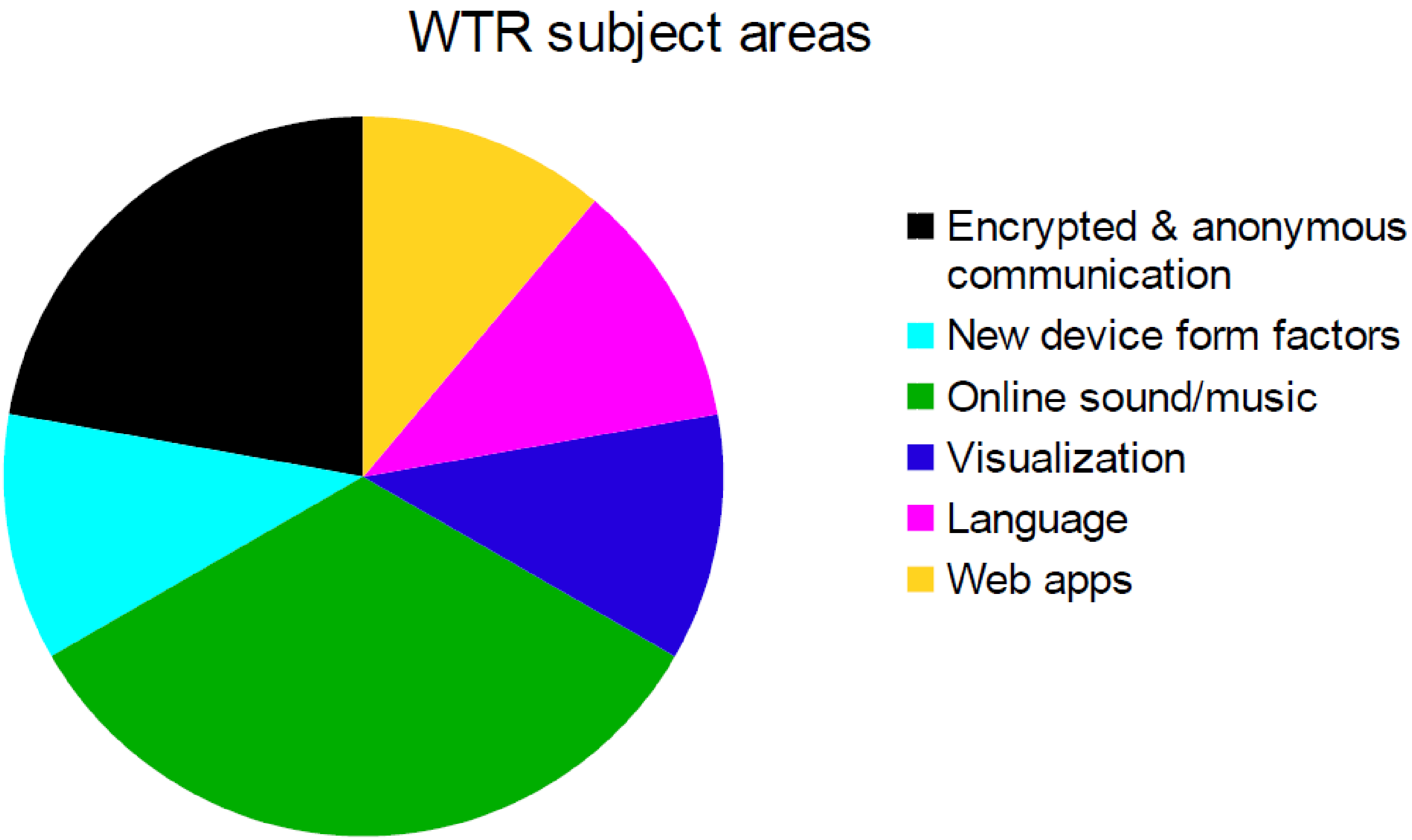
Staas de Jong

Universiteit Leiden > LIACS > Media Technology

Notes beforehand...

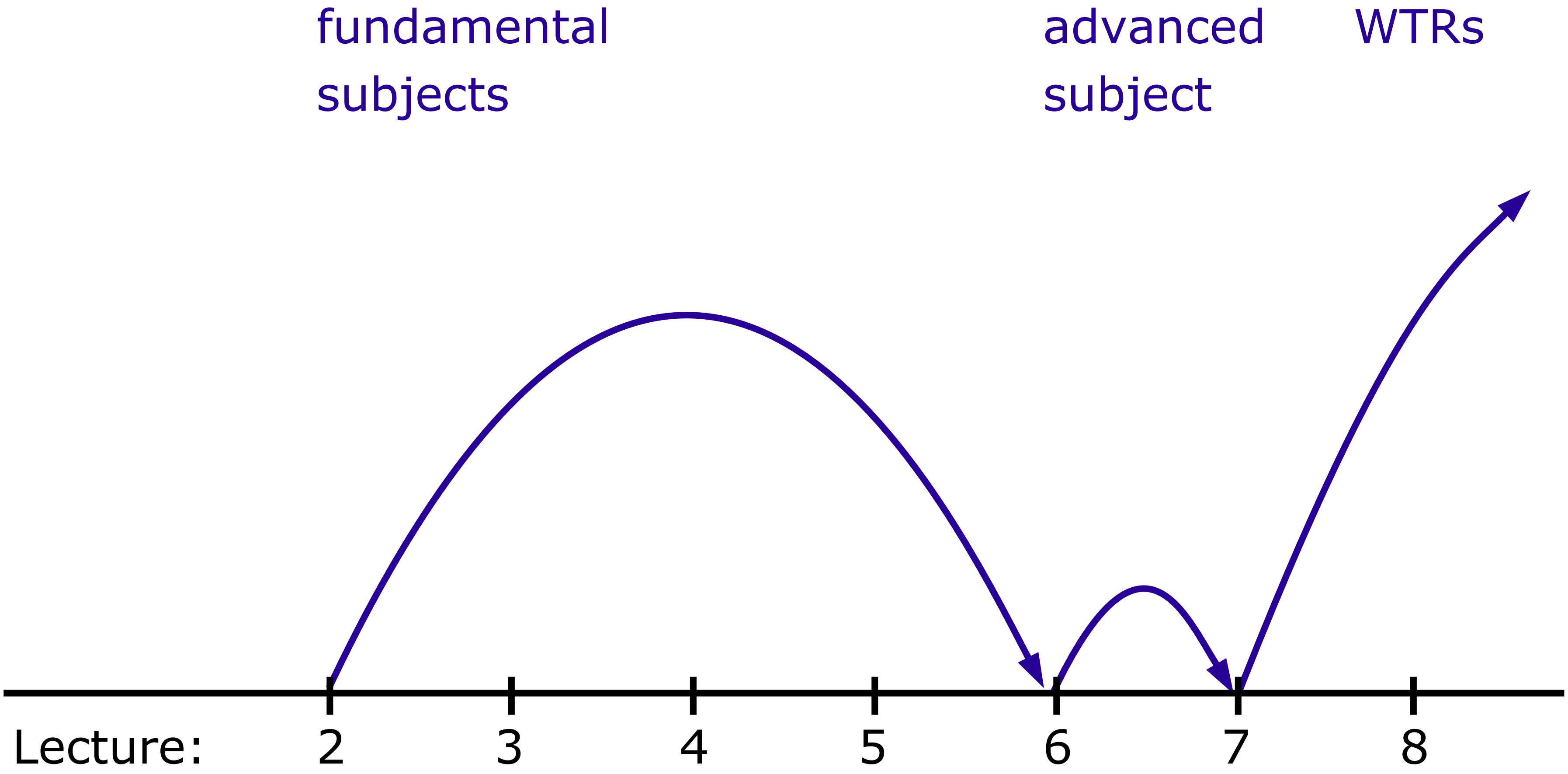


Notes beforehand...



↑ For more details: See the (online) presentation program.

Topical overview: main arcs



fundamental subjects

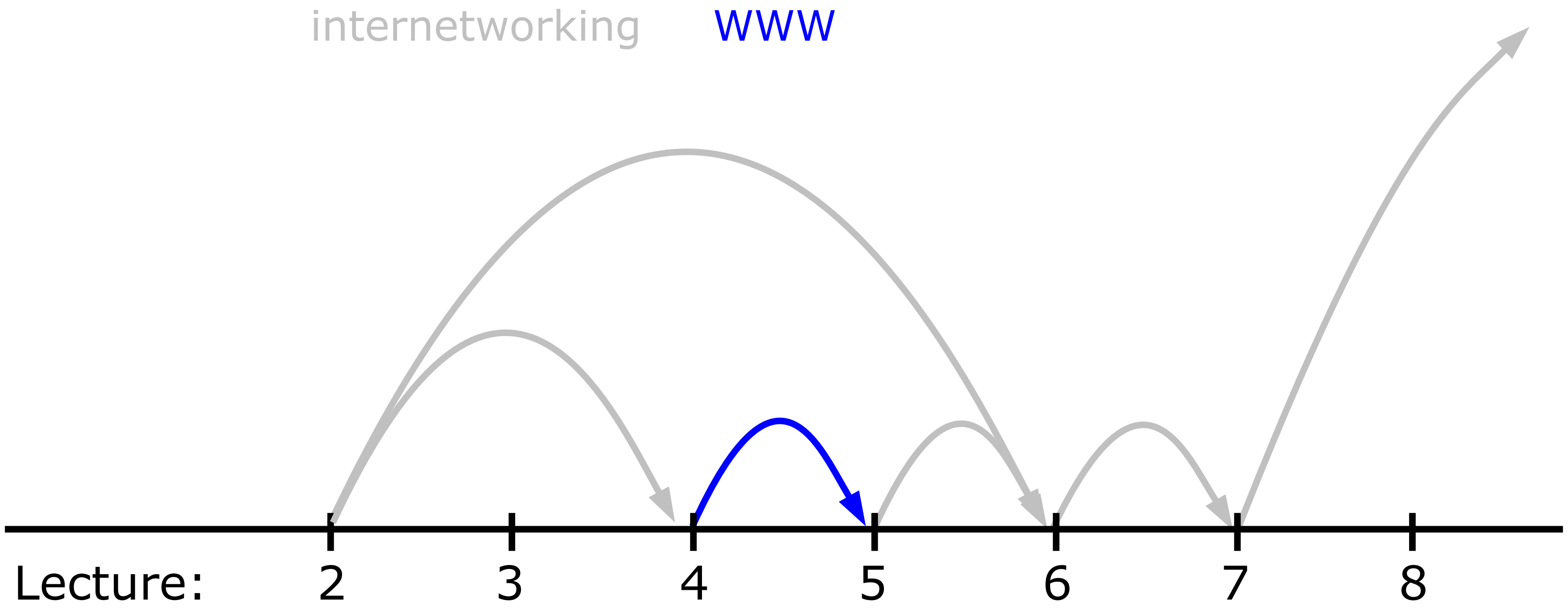
advanced subject

WTRs

Lecture: 2 3 4 5 6 7 8

Today: the *WWW* arc

- *Sessions 2-5:* from copper wires
  to client/server programming



The context

- *Old dream:* to store and make accessible all of human knowledge.

An innovative idea

- **“A *global* storage system where accessing one document also gives *immediate access* to *all the other documents that it refers to.*”**

↑ *Compare against:* printed text, containing textual references,
stored in libraries.

- Hyperlink: immediate link from one document to another.

- Hypertext: text that contains hyperlinks.

- *Extension:* documents may contain hypermedia
– not just written text...

Implementing hypertext, a “shopping list”:

- a format for storing hypertext documents

  including a uniform notation for hyperlinks

- devices that then store & instantly serve hypertext documents

- devices that can instantly retrieve & display hypertext documents

⇒ a communication protocol for transferring hypertext documents

- an open-ended mechanism extending hypertext with hypermedia

Implementing hypertext?

- *For most of human history:* at most, a far-away dream.

- *Since quite recently:* implementation has become technologically possible.

    - *Consider the gains discussed in the previous 2 lectures:*

      · IP · DNS · TCP · client/server technology

    ⇒ ...we can do this!

Implementing hypertext

- a format for storing hypertext documents

  ⇒ *HTML: HyperText Markup Language*

  including a uniform notation for hyperlinks

  ⇒ *URL: Uniform Resource Locator*

- devices that then store & instantly serve hypertext documents

  ⇒ *web servers*

- devices that can instantly retrieve & display hypertext documents

  ⇒ *web clients* = *browsers*

⇒ a communication protocol for transferring hypertext documents

  ⇒ *HTTP: HyperText Transfer Protocol*

- an open-ended mechanism extending hypertext with hypermedia

  ⇒ *MIME types*

Hyperlinks: URLs

- URLs are used as the identifier or address of some resource.
- “Resource”: e.g. a webpage, an image, a sound file, …
- A URL consists of a ‘scheme’ or ‘protocol’ to use (usually HTTP); a hostname; and a pathname.
    - e.g. http://www.liacs.nl/index.html
- Sometimes the TCP port number is also included.
    - e.g. as in http://www.liacs.nl:80/index.html
- URLs can be relative to the document they are mentioned in.
    - e.g., a mention of /edu/index.html may be short for http://www.liacs.nl:80/edu/index.html

URLs: examples of different protocols

web-resource	http://mediatechnology.liacs.nl
remote login (telnet)	telnet:krypton.wi.leidenuniv.nl
file transfer (ftp)	ftp://ftp.cs.uu.nl/pub/
Usenet newsgroups	news:comp.lang.javascript
e-mail	mailto:someone@liacs.nl
local file	file:c:\temp\mypage.html

Hypertext: HTML

- “Mark-up” in general: a notation used to specify how text should be displayed.

- Intended purpose of HTML markup:

  - specifying the *structure* of a hypertext document

  - *not* its *presentation*.

- HTML is strictly defined by the WWW Consortium (W3C): see http://www.w3.org.

HTML: hypertext markup

- Markup includes *tags*, *attributes*, and *entity references.*
- *Tags* are written `<x>` and `</x>`, where x is the tag name.
- Tags specify a specific markup, e.g. `<p>` for paragraphs, `<h1>` for headings, `<a>` for "anchors" = hyperlinks.
- *Attributes* specify additional parameters to tags, e.g. `href` inside an `<a>` tag, for the specific target URL of the hyperlink.
- Entity references encode special characters, and are written between '`&`' and '`;`' characters, e.g. `&gt;` for character '`>`' .
- Combined example: `<a href="index.html">&gt;&gt;</a>`

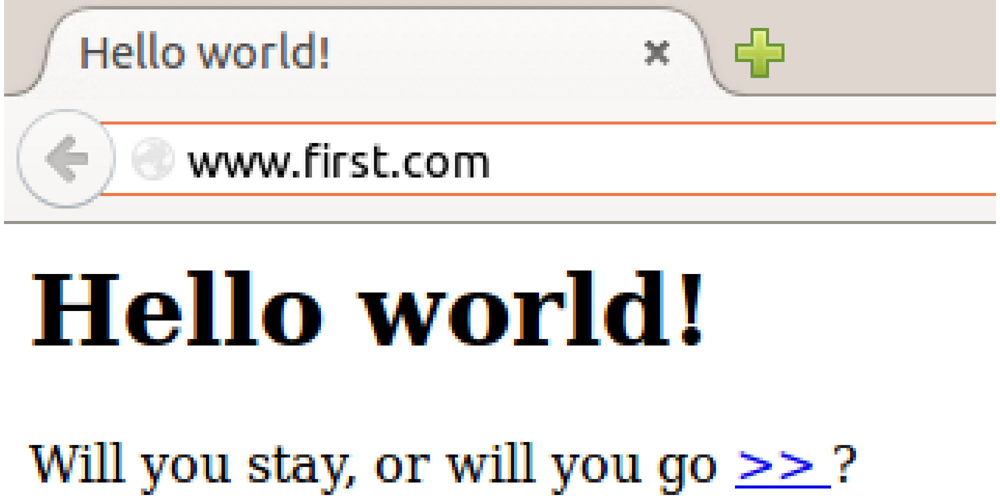
HTML: structured, navigable hypertext

A complete example – *HTML in a plaintext editor:*

```
<html>

  <head>
    <title> Hello world! </title>
  </head>

  <body>
    <h1> Hello world! </h1>
    <p>
      Will you stay, or will you go
      <a href="http://www.next.com"> &gt;&gt; </a> ?
    </p>
  </body>

</html>
```

HTML: structured, navigable hypertext

A complete example – *displayed by a browser:*



Hello world!

Will you stay, or will you go >> ?

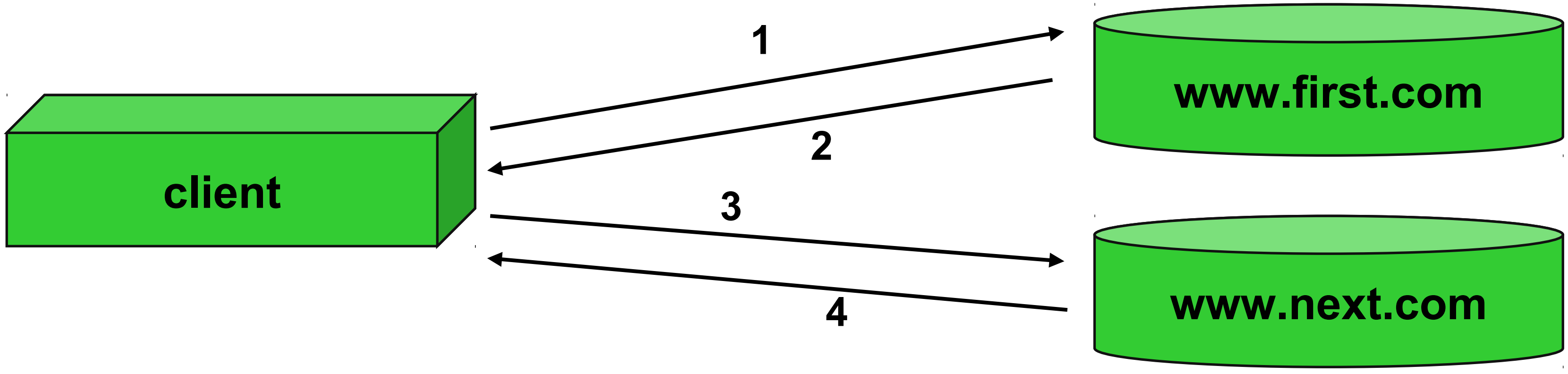
Navigation: *enabled* by HTTP transactions

- The user typed the URL of the example webpage (say, *http://www.first.com*) into the browser.

- The browser then sent a request to the appropriate server.

- The server responded with an HTML page.

- The user may click the hyperlink on the page, triggering a new request-response cycle, involving another machine:



HTTP transactions

- HTTP: *application layer protocol*.

    ↑ HTTP runs on top of TCP (which runs on top of IP).

    · HTTP servers: by default on TCP port 80.

- For transfer of HTML documents between web servers and web clients (browsers).

- Also used for transfer of other document/data types.

HTTP transactions

Classical HTTP scenario:

- client browser connects to server, using TCP;
- client sends request to server, using HTTP;
- server replies with a response, using HTTP;
- server disconnects the TCP connection.

HTTP/0.9 transactions

- HTTP/0.9: first official version (Berners-Lee 1991).
- Very basic:
  - client only has GET “request method” – *nothing else*
  - server simply responds with HTML content – *nothing else*.
- *An example client request:*

```
GET /~user/WebTech/
```

- *An example server response:*

```
<h1>Web Technology</h1>
<h2>Introduction</h2>
<ul>
<li><a href="Day1/internet.htm">Slides</a>
<li><a href="Day1/tutorial.html">Tutorial</a>
</ul>
 …
```

HTTP/1.0 transactions: more *request methods*

- **`GET`**: retrieve a document.
- **`HEAD`**: retrieve information about the document, but not the document itself.
- **`POST`**: provide information to the server.
- **`PUT`**: provide a new or replacement document to be stored on the server.
- **`DELETE`**: remove a document from the server.
- **`TRACE`**: ask that proxies declare themselves (in the headers, see below), so client can learn path taken by document.
- **`OPTIONS`**: what other methods can be used?

HTTP/1.0 transactions: *requests/responses*

An HTTP/1.0 *request* contains:

- a request method (usually `GET` – retrieve a document);
- a URL, identifying the document to be retrieved;
- an HTTP version number: `HTTP/1.0`;
- additional information in header lines;
- an empty line;
- optionally, a request body (when request method is `POST`).

An HTTP/1.0 *response* then contains:

- an HTTP version number: `HTTP/1.0`;
- a status code (e.g. “200”) indicating success or failure, and a textual annotation (e.g. “`OK`”);
- additional information in header lines;
- an empty line;
- a response body: the data to be retrieved.

HTTP/1.0 transactions: *status codes*

- ...are organized in ranges.

Codes:	**Meaning of the response:**
100-199	informational (e.g. Continue, Switching protocols);
200-299	client request successful;
300-399	client request redirected, further action necessary;
400-499	client request incomplete;
500-599	server errors.

- Most well-known are “200 OK” and “404 Not found”.

HTTP/1.1 transactions

- HTTP/1.1: currently the commonly used version.

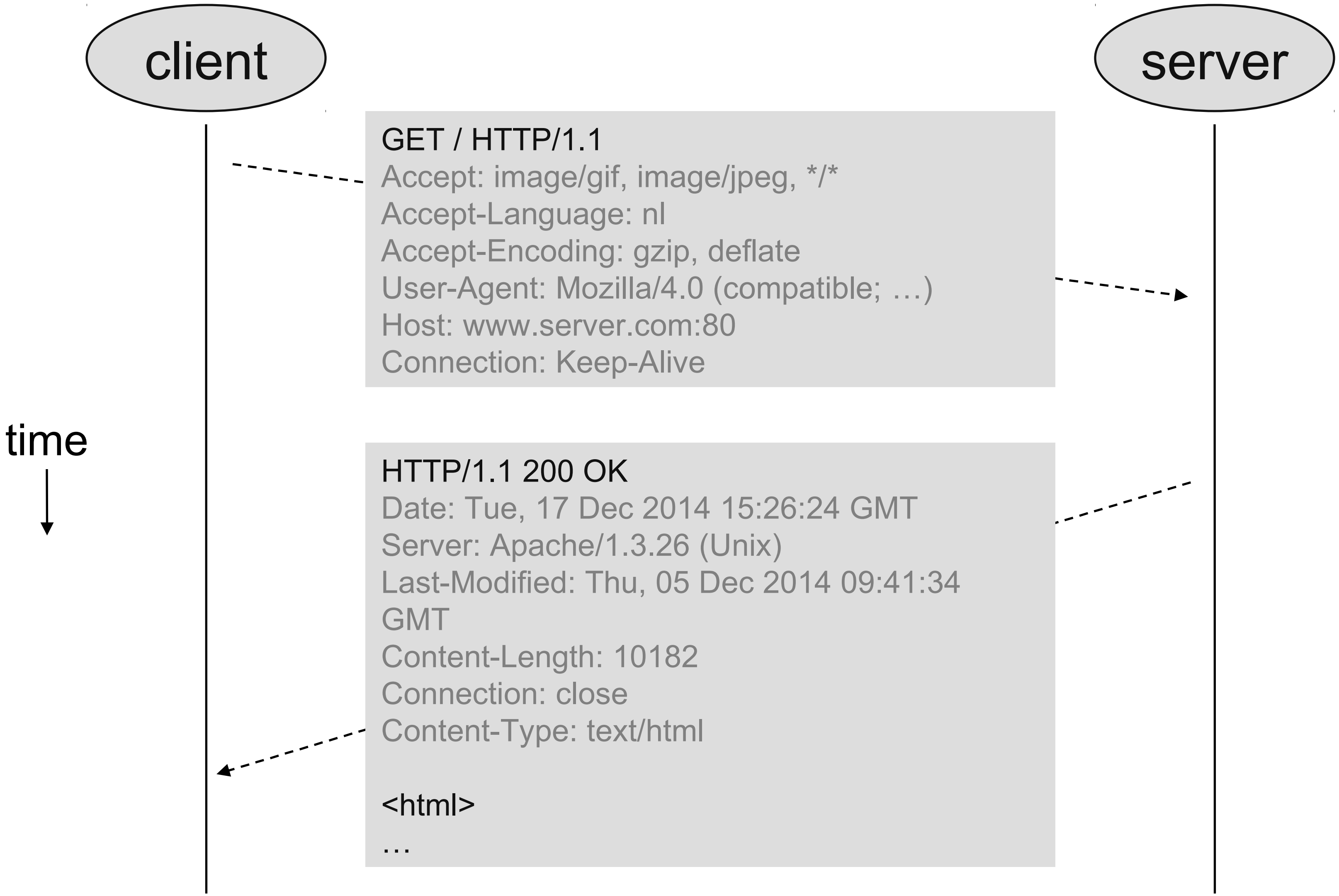
- Works much the same as HTTP/1.0.

  ⇒ Most important difference:

  - **HTTP/1.0:**
    - For each request-response transaction, there is a separate TCP connection to (the same) web server.

  - **HTTP/1.1:**
    - TCP connection is reused multiple times, e.g. to download images for a just-delivered page (persistent connections).

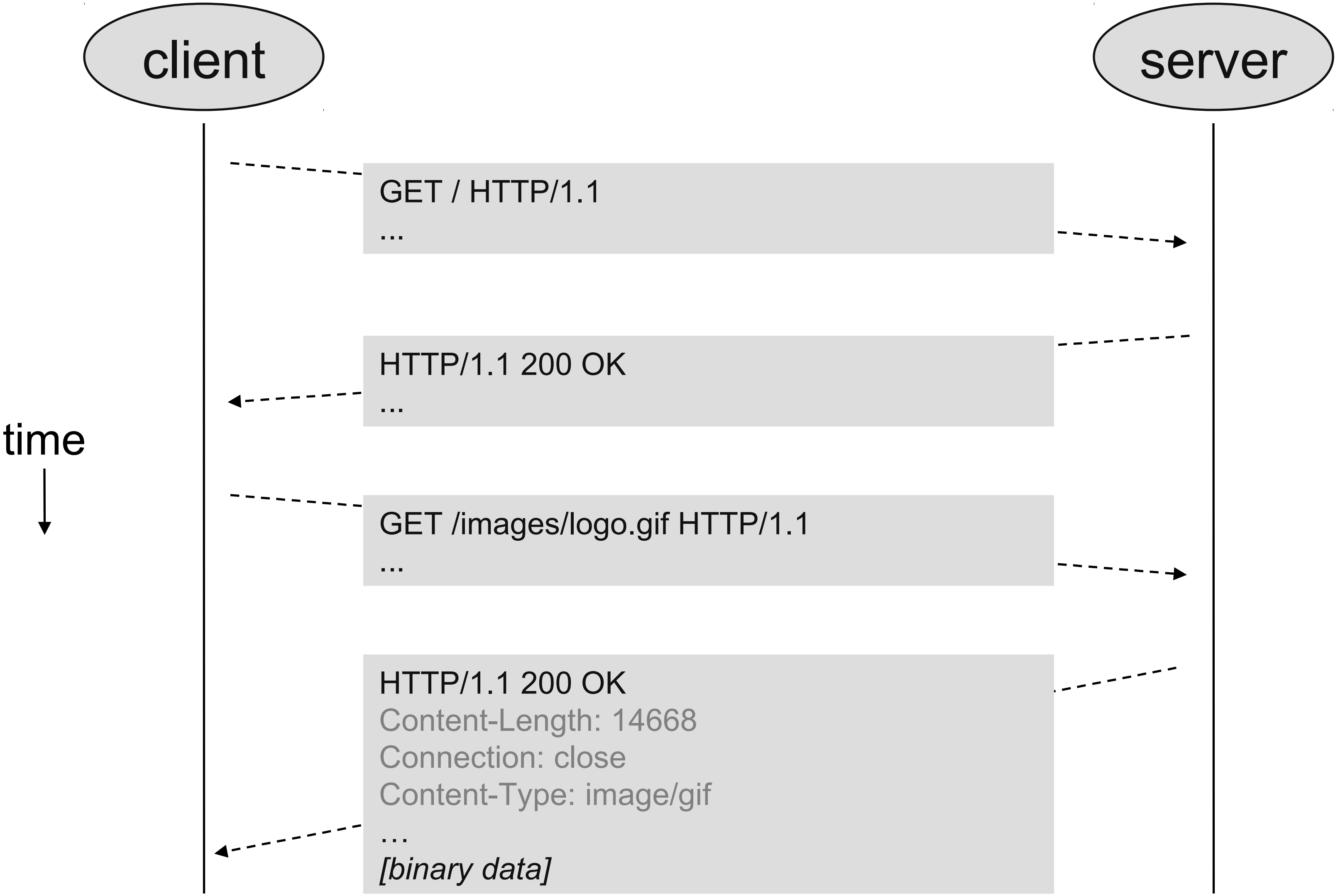
HTTP/1.1 transaction: a document



```
GET / HTTP/1.1
Accept: image/gif, image/jpeg, */*
Accept-Language: nl
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; …)
Host: www.server.com:80
Connection: Keep-Alive
```

```
HTTP/1.1 200 OK
Date: Tue, 17 Dec 2014 15:26:24 GMT
Server: Apache/1.3.26 (Unix)
Last-Modified: Thu, 05 Dec 2014 09:41:34
GMT
Content-Length: 10182
Connection: close
Content-Type: text/html

<html>
…
```

HTTP/1.1 transaction: a document with an image



client

server

time

```
GET / HTTP/1.1
...
```

```
HTTP/1.1 200 OK
...
```

```
GET /images/logo.gif HTTP/1.1
...
```

```
HTTP/1.1 200 OK
Content-Length: 14668
Connection: close
Content-Type: image/gif
…
[binary data]
```

BREAK!