Universidad
Politécnica
de Cartagena

Applications on the Internet

Lesson 2. Fundamentals of the Web. The Static Web

Goals

At the end of the lesson the student should be able to

- *Explain the operation of the HTTP protocol and its forms of use to get web resources.*
- *Describe the parts of a URL, its utility and disadvantages.*
- *Recognize a resource on the Web and describe different forms of representation the resources of the Web.*
- *Explain the basic operation of a HTTP server.*
- *Explain the basic operation of a browser.*

Contents

1. Introduction
2. Identifiers
    1. URL.
    2. URN.
3. HTTP protocol.
    1. Request format.
    2. Response format.
    3. Headers.
    4. Connections.
    5. State information.
    6. HTTP 2.0
4. Representation of resources: HTML.
    1. Relative and absolute URLs.
5. The browser.
6. HTTP servers.
7. Quiz.
8. Bibliography and extras.

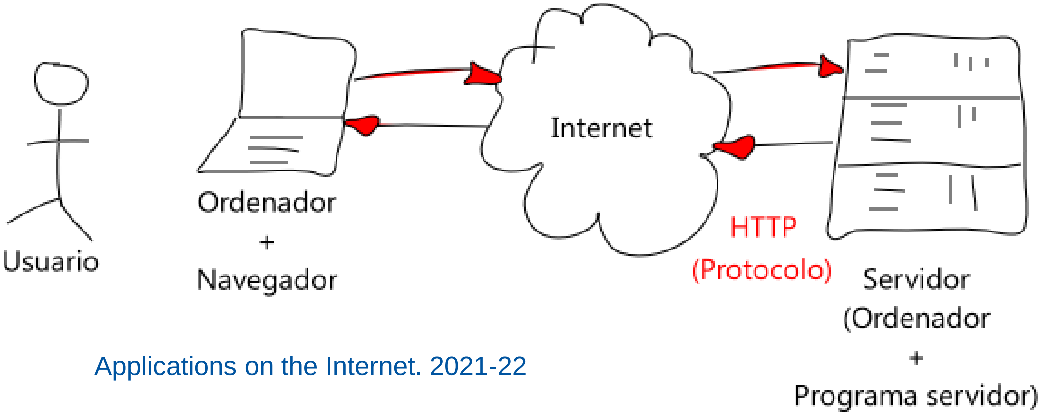
Introduction

Definitions:

- World Wide Web: a service to access, via Internet, to *hyperlinked* documents / resources. "
- The Web as a system is a set of technologies used to access linked resources.

Based on three simple technologies whose goals are respectively:

- Identify resources (URI)
- Represent resources (HTML and others)
- Transfer resources (HTTP).



Contents

1. Introduction
2. **Identifiers**
    1. URL.
    2. URN.
3. HTTP protocol.
    1. Request format.
    2. Response format.
    3. Headers.
    4. Connections.
    5. State information.
    6. HTTP 2.0
4. Representation of resources: HTML.
    1. Relative and absolute URLs.
5. The browser.
6. HTTP servers.
7. Quiz.
8. Bibliography and extras.

Identifiers

Arise from the need to have a mechanism for naming and locating resources in a standard way. Such a mechanism should address:

- How is the resource **named**?
- **Where** is the resource?
- **How** you can **access** the resource?

Definition: A URI (*Uniform Resource Identifier*) is a character string that identifies resources on the Web.

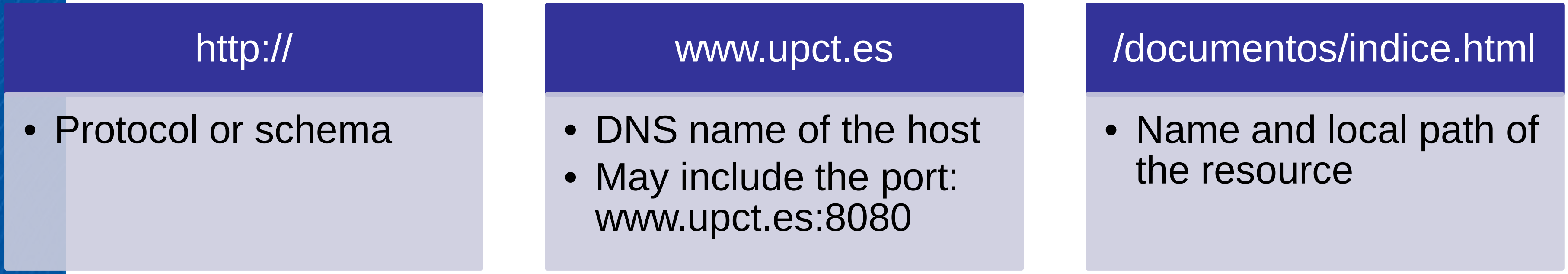
- The string has a very simple syntax and can be arbitrarily long.
- There are two types of URI:
  - Uniform Resource Locator (URL): indicates the location of the resource and its form of access.
  - Uniform Resource Name (URN): uniquely identifies a resource, but not its location or how to access it.

Identifiers (URL)

Uniform Resource Locator (URL)

- They have 3 parts:

http://www.upct.es/documentos/indice.html



http://

- Protocol or schema

www.upct.es

- DNS name of the host
- May include the port: www.upct.es:8080

/documentos/indice.html

- Name and local path of the resource

- **Protocol** or schema: describes how to access the resource. HTTP is the standard protocol for web resources but there are others.
- **Address:** Indicates an IP address or domain name of the machine that stores the resource. You can optionally include the port.
- **Local path:** Identifies the local path inside the machine where the resource is stored
- **Query string**: Additionally may include an information string as a list of variable = value pairs sent to the server. For example: http://ait.upct.es/usuario.php**?age=24&sex=male**

Identifiers (URL)

Examples:

ftp://example.org/resource.txt

ldap: //ldap.example.com/dc=example,dc=com

telnet: //192.0.2.16: 80 /

mailto: John.Doe@example.com

file: /// C: /Users/eegea/Documents/index.docx

file: /// etc / exports

Schema name	Description	Reference
ftp	File Transfer Protocol	RFC 1738
http	Hypertext Transfer Protocol	RFC 2616
mailto	Electronic mail address	RFC 2368
file	Local file access	RFC 1738
ldap	lighweight Directory Access Protocol	RFC 4511

Identifiers (URL)

Not all the URLs use the format protocol, address and local path, although the most used on the Internet do.

To dereference a URL: **It is the process that an application (usually a browser) perform to access the resource identified. It depends on the type of scheme.**

- In the case of HTTP: to establish a TCP / IP connection to the machine indicated in the address and to send an HTTP request.

URI can be used to define interfaces

- Used in web applications
- URI templates (*URI Templates, RFC 6570*)
- Used to declare functions and variables
- They are especially useful for automatic machine processing
- http://ait.upct.es/usuario/{id}
- http://ait.upct.es/usuario/1234

Identifiers (URN)

Uniform Resource Name (URN)

- Character string that uniquely identifies a resource independently of its location.
- Its format is different from the URL: *urn:schema:nss*
- urn:isbn:0451450523

Example: magnet link

- *de facto* standard to declare content identifiers in P2P
- magnet:?xt= urn:sha1: YNCKHTQCWBTRNJIV4WNAE52SJUQCZO5C
- How do you know where the resource is, then?

Contents

1. Introduction
2. Identifiers
    1. URL.
    2. URN.
3. **HTTP protocol.**
    1. Request format.
    2. Response format.
    3. Headers.
    4. Connections.
    5. State information.
    6. HTTP 2.0
4. Representation of resources: HTML.
    1. Relative and absolute URLs.
5. The browser.
6. HTTP servers.
7. Quiz.
8. Bibliography and extras.

The HTTP protocol. characteristics

HyperText Transfer Protocol

- Protocol for transferring resources on the Web. Specified in RFC 2616.
- **Application level** protocol**.**
  - It is encapsulated over TCP/IP
- **Request / response** protocol.
  - It does not maintain state information between requests.
  - By contrast, TCP for example, does maintain state information: LISTEN, ESTABLISHED ...
- **Text-based** protocol.
  - It does not define a sequence of fields of predetermined length, but the information is encoded as a string of ASCII characters (7 bits).
- Uses the **port 80** as default.

HTTP. Request format

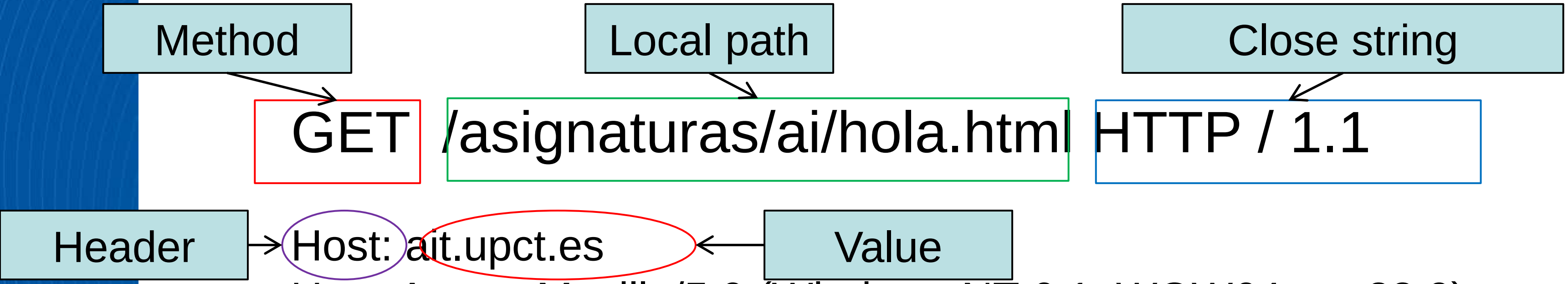
The *request message* is a *character string* consisting of:

- METHOD + LOCAL PATH + CLOSE string

Followed by one or more *headers* (also text strings) in the form:

- Field: Value

http://ait.upct.es/asignaturas/ai/hola.html



```
GET /asignaturas/ai/hola.html HTTP / 1.1
Host: ait.upct.es
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv: 23.0)
Gecko/ 20100101 Firefox/23.0
Accept* / *
Accept-Language: in-US, in, q= 0.5
Accept-Encoding: gzip, deflate
Connection: keepalive
```

HTTP. Request format

Method: Operation to perform

- Declares the **type of request** which it is performed. Each operation would be the equivalent of a different "packet type" in low-level protocols.
- The methods define the **protocol interface :** define all types of permitted operations.
- Commonly used methods are:

Method	**Description**
GET	Gets a resource
HEAD	Gets the resource metadata
PUT	Updates or alters a resource
POST	Implies creating a new resource
DELETE	Remove resource
OPTIONS	Asks the methods available for a resource

HTTP. Request format

***Local path* identifies the requested resource within the host machine**

- It corresponds to local path segment of the URL.

The server concatenates the local path to its *document root directory* and form the absolute path inside the machine.

- Example: http://ait.upct.es/asignaturas/ai/ad/index.html
  - In ait.upct.es the root directory is /var/www
  - The absolute path is /var/www/asignaturas/ai/ad/index.html
- Only documents under the document root directory can be served.
- If a file is not specified in the URL, the server returns a default document, for example, index.html. If it does not exist, a file list is returned. Or an error is returned. These options can be configured on the server.
- When request comes without a local path, it is requesting the default document root. Example,http://www.google.com It translates into a request GET / HTTP/1.1

***Close string* indicates the protocol version.**

- HTTP/1.0 First version of the protocol.
- HTTP/1.1 Most used.
- HTTP/2.0 Current version. Published in 2015 (RFC 7540).

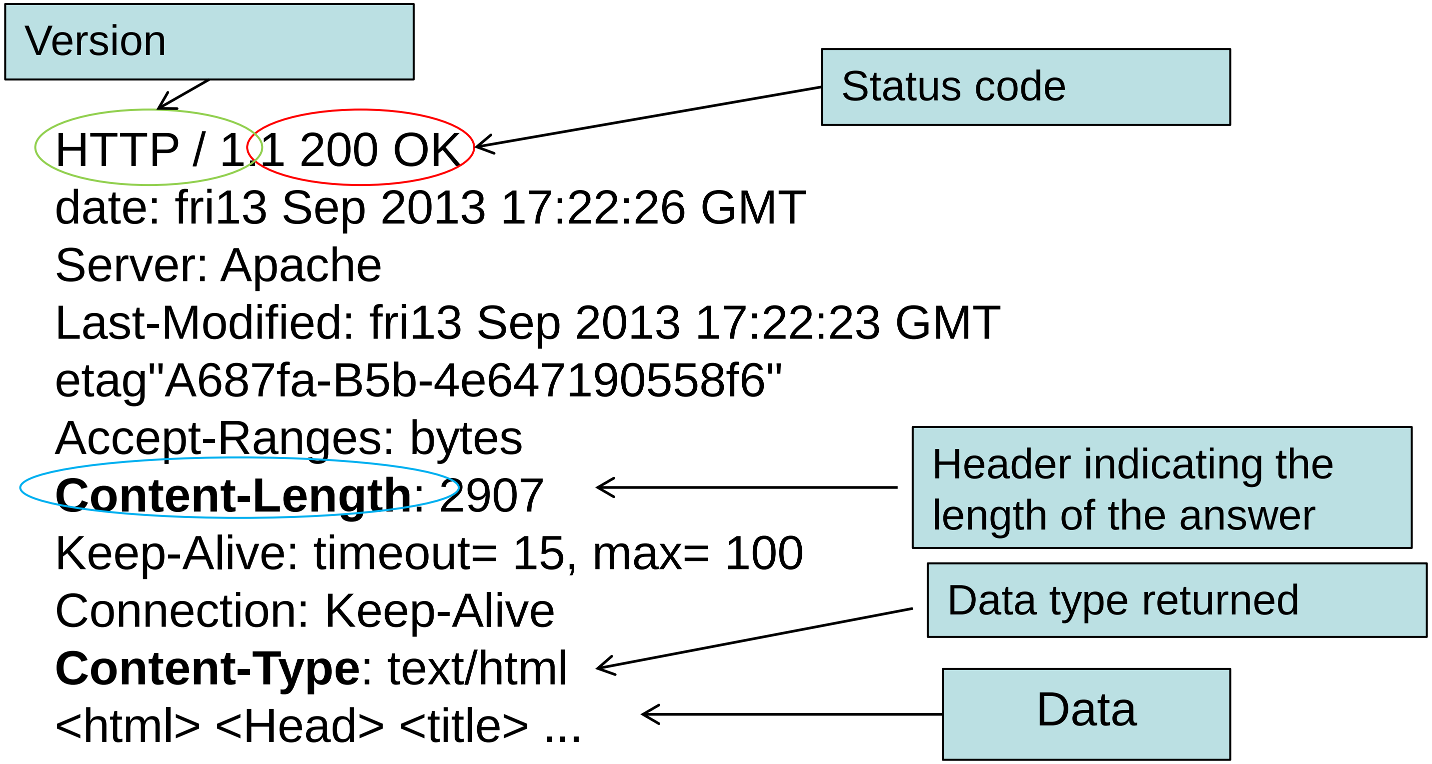
Activity 1. HTTP Requests

- **http://map.gsfc.nasa.gov/index.html**
- **http://map.gsfc.nasa.gov/**
  - Is there a difference between requests? Do you get the same data?
  - Indicate that URL will be asked if you click on "WMAP Technical Papers".
  - Click this URL and observe the result.
  - If the root directory in map.gsfc.nasa.gov is /var/www/ Indicate the absolute path inside the machine for some of the available links.
  - Navigate to the image gallery. Enter the URLs of some of the images available.

HTTP. Response format

The response message is a string with headers in addition to the requested data

- Version + status code
- Response Headers
- Response data (requested resource).

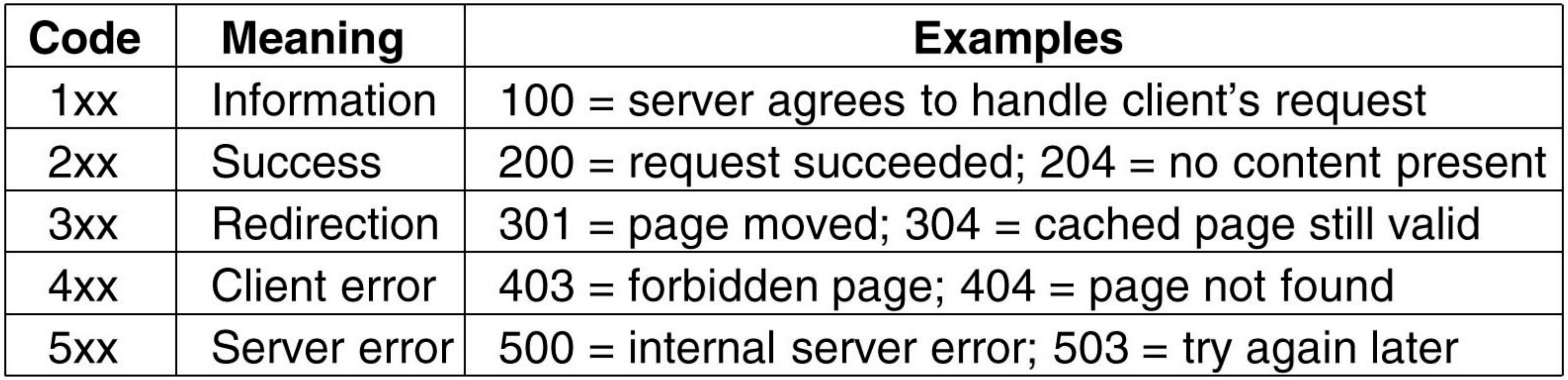


```
HTTP / 1.1 200 OK
date: fri13 Sep 2013 17:22:26 GMT
Server: Apache
Last-Modified: fri13 Sep 2013 17:22:23 GMT
etag"A687fa-B5b-4e647190558f6"
Accept-Ranges: bytes
Content-Length: 2907
Keep-Alive: timeout= 15, max= 100
Connection: Keep-Alive
Content-Type: text/html
<html> <Head> <title> ...
```

HTTP. Response format

Status code

- It indicates the result of the operation.
- Example: 200 OK. correct request.
- Example: 304 not modified. The server indicates that the document was recently requested and that has not changed so it is not forwarded. The client must have it cached.



Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client’s request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

HTTP. Headers

Both the request and the response may include pairs of *fields: value* additional information

- Multiple headers are specified: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

There are no restrictions on the type of resources that is supported on the web. They can be audio, video, text, executable ...

Content-length header:

- Size in bytes of the data contained in the message.

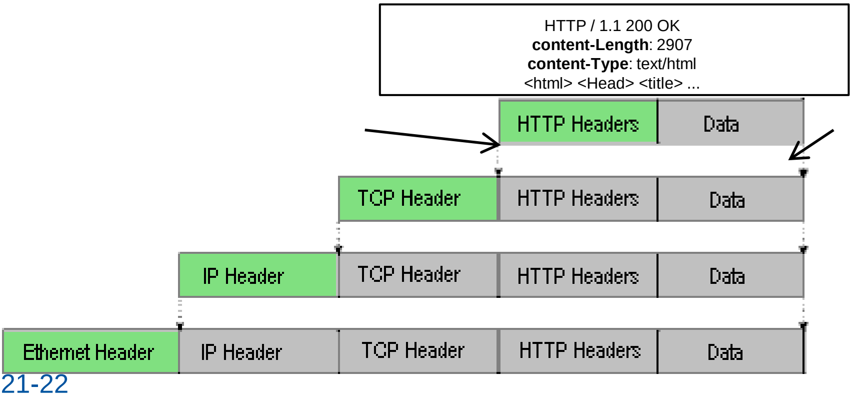
Content-type header:

- It indicates the type of data that is returned.
- It is indicated by the **MIME type or Media Internet Types**.
  - They are standard strings that identify data formats
  - The format is *type/subtype*
  - Examples: text/html   image/jpeg   application/zip

HTTP. Connections

HTTP is encapsulated over TCP/IP.

- Requests will default to port 80.
- The request message is encapsulated over a TCP connection to the server.
- The server uses the TCP connection to return the reply message with the data.
- However, the requested resource in most cases is composed of multiple elements. For example, HTML document with images.
- The client receives the HTML document and process it to see if more ítems are needed to display the document.
- With **HTTP 1.0** It was necessary to establish **an additional TCP connection for each element.**
  - very inefficient
- With **HTTP 1.1 All necessary elements are requested on the same TCP connection. Parallel connections are established.**
  - Use the header Connection: Keepalive

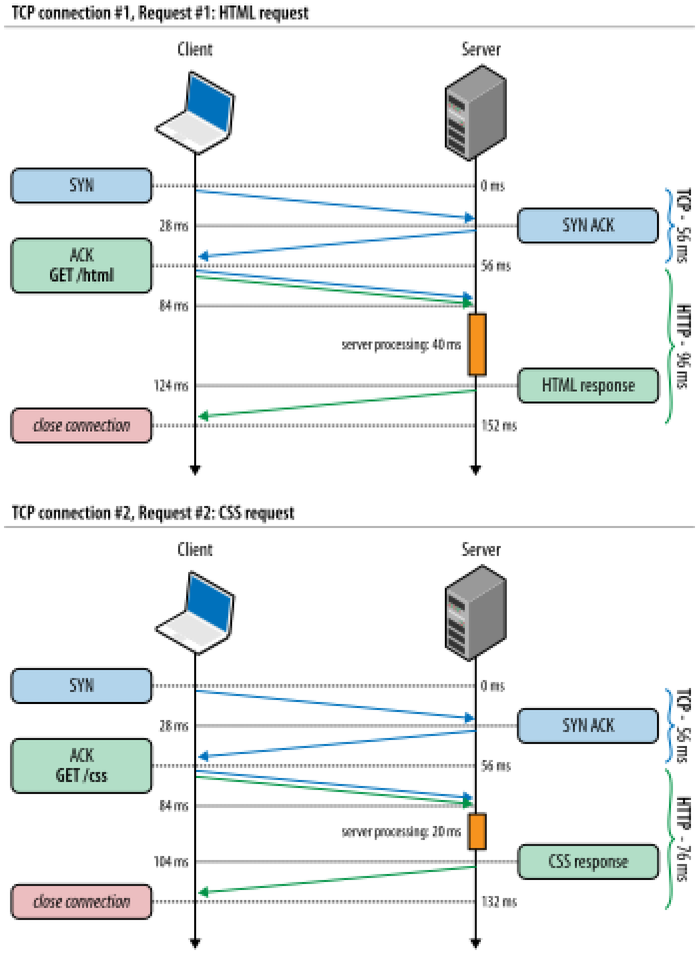


HTTP / 1.1 200 OK
content-Length: 2907
content-Type: text/html
<html> <Head> <title> ...

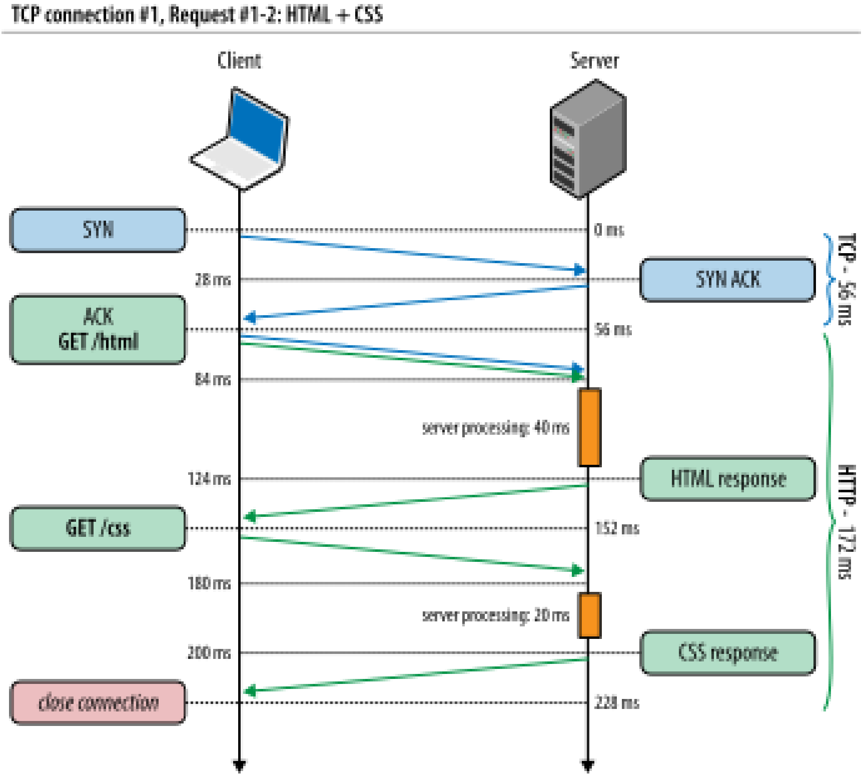
HTTP Headers | Data
TCP Header | HTTP Headers | Data
IP Header | TCP Header | HTTP Headers | Data
Ethernet Header | IP Header | TCP Header | HTTP Headers | Data

HTTP 1.0 and 1.1 connection examples

HTTP 1.0: 2 connections for HTML and CSS



HTTP 1.1: 1 connection for both HTML and CSS



Activity 2. Connections and encapsulation

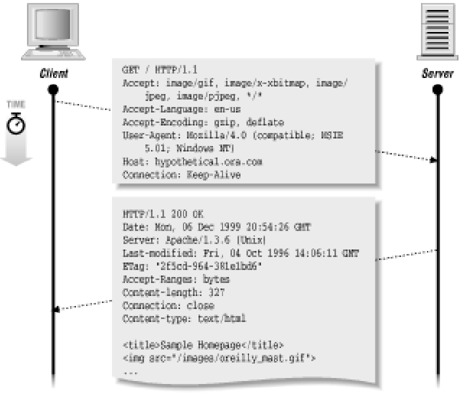
http://labit601.upct.es/~eegea/

- run wireshark and capture exchanged messages.
- Examine the encapsulation of the petition and original response.
- How many additional elements are needed to display the page?
- How many TCP connections are established with the server?
- With development tools browser, recharge page and examine the "Net" tab
- Repeat the exercise for http://ait.upct.es/asignaturas/ai/con.html
- How many TCP connections have been established in this case? Why?
- Examine the response requests images wiresharkWhere and in what format the data appear?
- Reload the page and examine the request headers and answer with development tools browser, What is the status code? what does it mean?

HTTP. Status Information

HTTP is a *request and response* protocol

- Requests are independent of each other.
- The server does not store any **status information** of the "conversation" between him and the client. It is not possible to know what a client has done ("who has visited") previously, or, in which state it is.
- **Advantage**: Is much more **scalable**. The server does not have to store status information.
- **Issues**: You can not make applications like "shopping cart" and many others requiring information on what the client had done before (why?).
  - **Solution:** Extension of the protocol and browser: **cookies**.



```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/
     jpeg, image/pjpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE
     5.01; Windows NT)
Host: hypothetical.ora.com
Connection: Keep-Alive
```

```
HTTP/1.1 200 OK
Date: Mon, 06 Dec 1999 20:54:26 GMT
Server: Apache/1.3.6 (Unix)
Last-modified: Fri, 04 Oct 1996 14:06:11 GMT
ETag: "2f5cd-964-381e1bd6"
Accept-Ranges: bytes
Content-length: 327
Connection: close
Content-type: text/html

<title>Sample Homepage</title>
<img src="/images/oreilly_mast.gif">
...
```

HTTP. Others

It is possibly the most widely used protocol at the application level.

- Usually is not filtered by the *firewall*.
- It is used in multiple contexts in addition to the Web.
- It supports almost any format, simply using proper MIME type in content-type.

It is relatively inefficient.

- Text-based.
- The content can be compressed.

HTTP is insecure.

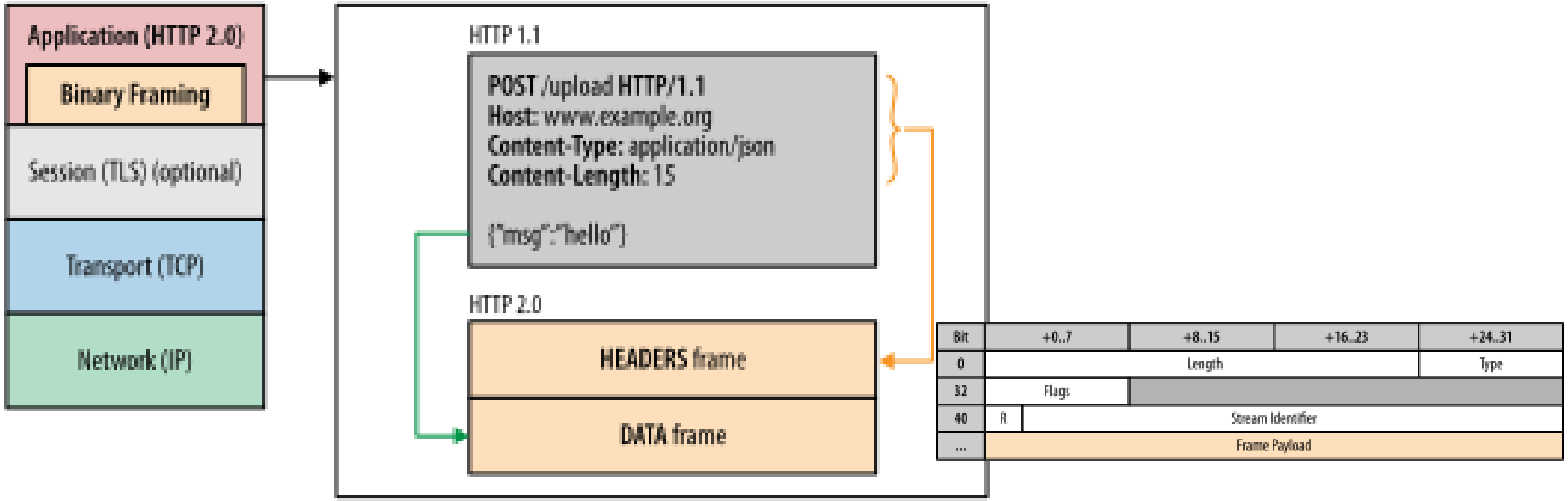
- To provide security used in combination with other techniques.
- Typically it used on a layer of safe transportation, as *Secure Sockets Layer* (SSL) or *Transport Layer Security*(TLS). In that case, the scheme is *https*.

HTTP 2.0

- **Arises from SPDY**
  - SPDY was an experimental protocol, developed at Google
  - The HTTP/2 standard is one of the most extensively tested standards right out of the gate.
- **Goal: improve the performance of HTTP**
  - Reduce latency
    - How: full request and response multiplexing
  - Reduce overhead:
    - How: efficient compression of HTTP header fields
  - support for request prioritization and server push
- **Extends (not replace) the previous standard**
  - The application semantics of HTTP are the same, and no changes to the offered functionality or core concepts such as HTTP methods, status codes, URIs, and header fields,
  - High-level API remains the same, the low-level changes

HTTP 2.0. Binary framing layer

- **Introduces a new optimized encoding mechanism between the socket interface (TCP) and the application layer HTTP**
  - Methods and headers, are unaffected, but the way they are encoded while in transit is
  - an HTTP/1.x client won't understand an HTTP/2 only server and vice versa

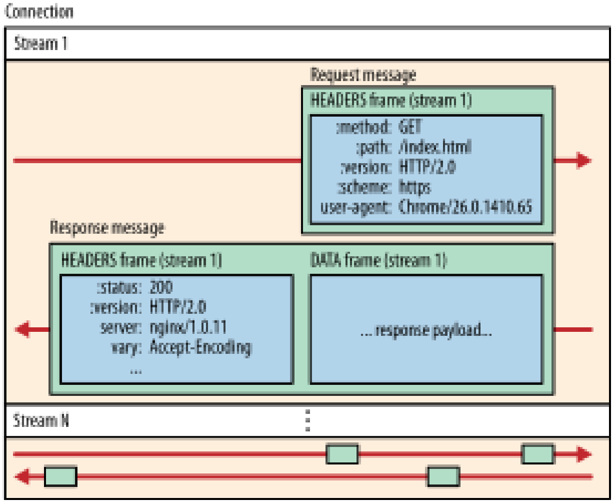


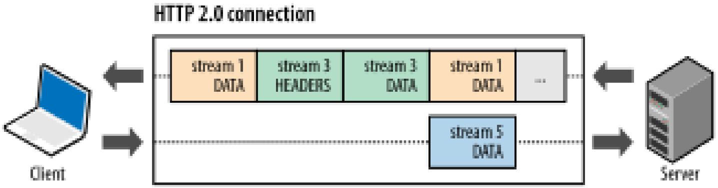
```
POST /upload HTTP/1.1
Host: www.example.org
Content-Type: application/json
Content-Length: 15

{"msg":"hello"}
```

Bit	+0..7	+8..15	+16..23	+24..31
0	Length			Type
32	Flags			
40	R Stream Identifier			
...	Frame Payload			

HTTP 2.0. Streams, messages and frames

- **Introduces streams, messages and frames**
  - Each message is a logical HTTP message, such as a request, or response, which consists of one or more frames.
- **Multiplexing of requests and responses**
  - Interleave multiple requests or replies in parallel without blocking on any one
  - Use a single connection to deliver multiple requests and responses in parallel





HTTP 2.0. Other features

- **Stream Prioritization**
  - A browser can communicate its priorities but it is decided by the server
- **One Connection Per Origin**
  - More efficient use of the TCP connection along all the path (user, router, server).
- **Support of flow control at application layer.**
- ***Server-Push (Data exchange initiated by server).***
  - Server decides sending data not requested explicitly
- **Header compression**

Contents

1. Introduction
2. Identifiers
    1. URL.
    2. URN.
3. HTTP protocol.
    1. Request format.
    2. Response format.
    3. Headers.
    4. Connections.
    5. State information.
    6. HTTP 2.0
4. **Representation of resources: HTML.**
    1. Relative and absolute URLs.
5. The browser.
6. HTTP servers.
7. Quiz.
8. Bibliography and extras.

Representation of resources: HTML

HyperText Markup Language (HTML)

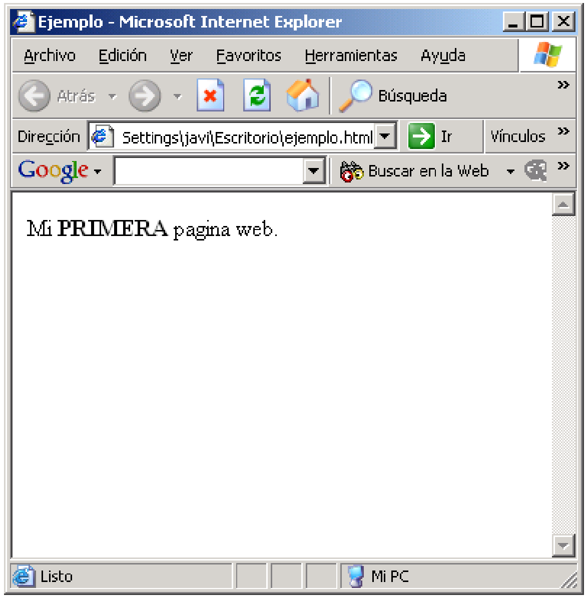
- Markup language used for representing resources on the Web.
- Is a **Text language**: Data is encoded as characters (ASCII, UTF-8, etc.).
- **Structured language**: Includes data and **metadata**, ie, "data about data".
  - **Markup Language**: Metadata included by **marks** or tags. Marks usually appear in pairs and the full pair is called an **HTML element**.
- An HTML document can include, as well as text, references to other formats (images, audio, video, etc.) to be displayed along with the rest of the content.
- An HTML document usually includes **links to** other resources and documents.
- An HTML document is also called *Web page*.

HTML elements

HTML elements are pairs of tags (between the "<" and ">") inserted in the text, whose meaning is interpreted by the browser.

- Examples: <h1> </ h1> <img> </img> <a> </a>
- The pair includes a start tag and an end tag (equal except it is preceded by "/").
- Elements may include attributes with pairs name = "value"

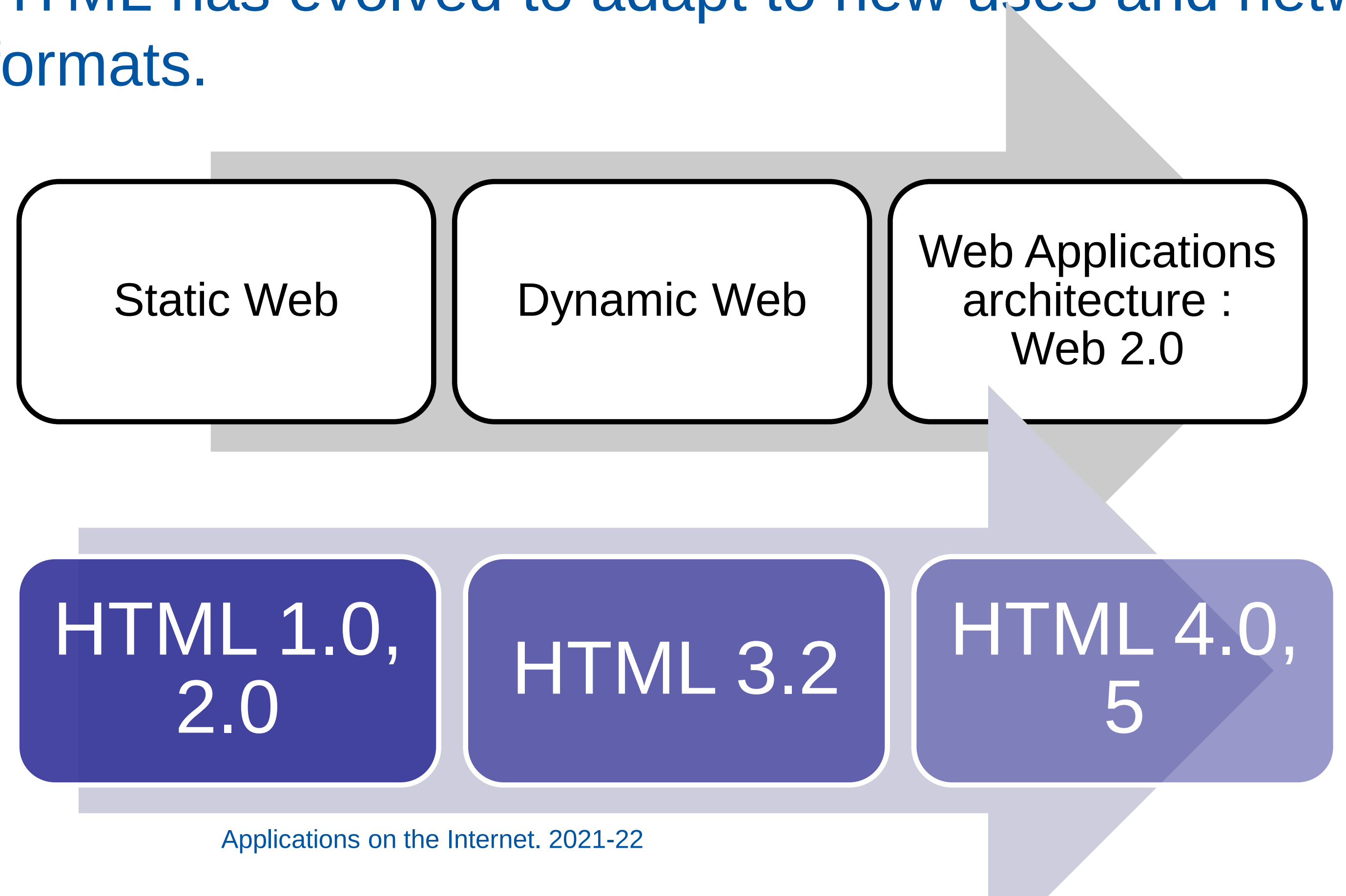
```
<HTML>

<HEAD>
<TITLE> Example </ TITLE>
</ HEAD>


<BODY>
My <B> first </ B> website.
</ BODY>


</ HTML>
```



Mi **PRIMERA** pagina web.

HTML evolution

- HTML in its early versions was focused on how to display the resources in a form intelligible to a person (*human-readable*).
- HTML has evolved to adapt to new uses and network formats.



Hyperlinks and relative and absolute URLs

Hyperlinks are represented in HTML by the <A> element (anchor or *anchor*).

- The referenced resource is indicated by the attribute *href* whose value is a URL.
- <a href= “http://ait.upct.es/index.html"> pincha</a>
- URLs can be **absolute** or **relative**
  - **Absolute ones** always include the domain name and schema.
  - The ***relative ones*** are relative to the current document, in fact, *relative to the directory where* the *current document is* .
  - The browser *solves* a relative URL and transforms it into an absolute before performing the request.

Relative URLs

Example: within an HTML document hosted in http://www.upct.es/ad/index.html appear the following URLs

- *images/a.gif* indicates that the picture is hosted in a subdirectory of the current URL, its absolute equivalent would be: http://www.upct.es/ad/imagenes/a.gif
- *../index.html*, its absolute equivalent would be http://www.upct.es/index.html
- *a.gif,* its absolute equivalent would be http://www.upct.es/ad/a.gif
- *./a.gif* its absolute equivalent would be http://www.upct.es/ad/a.gif
- */index.html* in this case the initial slash refers to the document root. Therefore, its absolute equivalent is http://www.upct.es/index.html

Activity 3. HTML and URLs

- http://labit601.upct.es/
  - Look at the *source code* this page (source document, HTML).
  - Identify different HTML elements.
  - Use development extensions of your browser to examine the *source code* of the page.
- www.firefox.com
  - Use the development extension *Net* from firefox to examine the requests that have been made to show this page and write the main request that has been made to the server.

Contents

1. Introduction
2. Identifiers
    1. URL.
    2. URN.
3. HTTP protocol.
    1. Request format.
    2. Response format.
    3. Headers.
    4. Connections.
    5. State information.
    6. HTTP 2.0
4. Representation of resources: HTML.
    1. Relative and absolute URLs.
5. **The browser.**
6. HTTP servers.
7. Quiz.
8. Bibliography and extras.

The browser

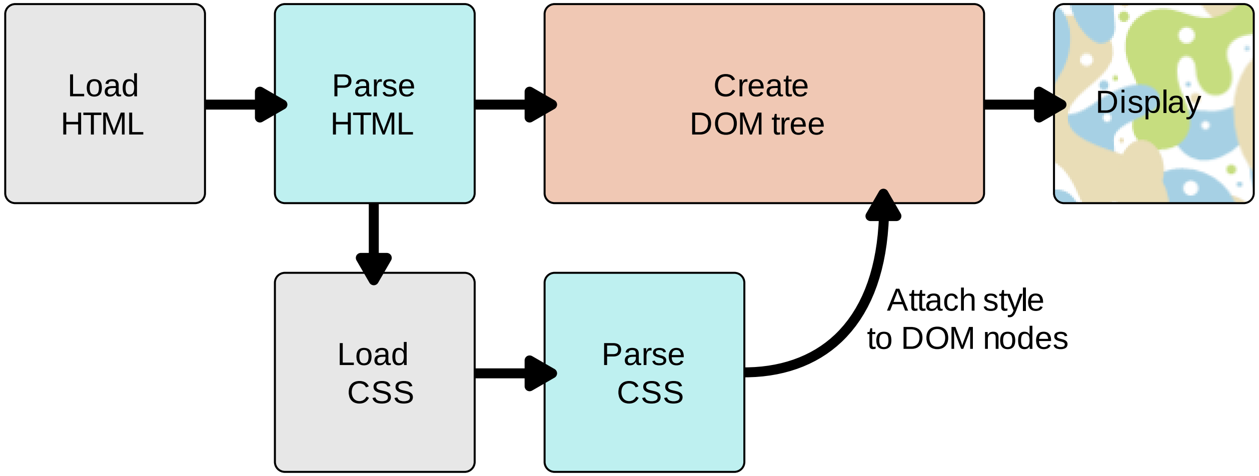
A program (application software) that implements an HTTP client and is capable of process and display HTML documents and other formats used as resources on the Web.

- **HTTP client.** Although most also implements other protocols (FTP, file...).
- **HTML parser**. HTML documents are processed and its contents displayed.
- There are many implementations: http://en.wikipedia.org/wiki/Comparison_of_web_browsers
- The browser can extend its functionality by :
  - The use of ***plug-ins.*** When the browser is unable to handle a format, it requests the use of an external program that it runs whenever a resource with that format is received. Examples: long time ago, PDF documents were displayed by Adobe Reader, video formats.
  - It allows the quick introduction of new formats and content on the web. Progressively incorporated into the browser.

Browser components and operation

- The browser displays the contents of HTML and CSS according to W3C specifications.
- Browsers are usually implemented by more or less independent functional blocks:
  - Display engine (*rendering/layout engine*): Is responsible for analyzing the content of the document and display it.
  - Motor connection (*networking engine*) Handles requests over the network.
  - Javascript interpreter (*javascript engine*): Responsible for locally executing the Javascript code.
  - User Interface.
- Flow of execution:
  - The rendering engine collects the contents of the document requested by engine connection (in 8KB blocks for example), parses the HTML and generates a DOM tree, from which it generates a tree display, assigns coordinates to the elements to be shown and finally render the contenn on the screen.
  - It is a gradual process, items are displayed as they are available
- Its performance determines the user experience.
  - Most of the "loading“ time is due to the representation of screen elements by the browser, not by the network delay.

Browser operation



Contents

1. Introduction
2. Identifiers
    1. URL.
    2. URN.
3. HTTP protocol.
    1. Request format.
    2. Response format.
    3. Headers.
    4. Connections.
    5. State information.
    6. HTTP 2.0
4. Representation of resources: HTML.
    1. Relative and absolute URLs.
5. The browser.
6. **HTTP servers.**
7. Quiz.
8. Bibliography and extras.

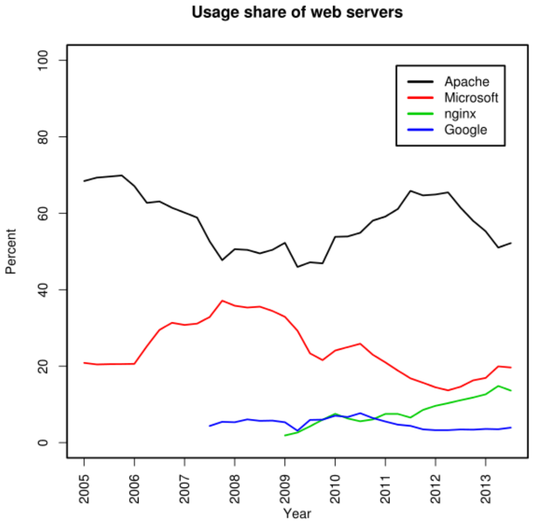
HTTP servers

Server

- A computer running a HTTP server which stores documents/files/resources that can be served via HTTP.

HTTP server

- A program (application software) serving requests made with the HTTP protocol.
- Basic Operation:
  - the server generates the absolute local path of the requested resource
  - access to that path in its local file system: if the file exists and it has permission, opens it and sends it to the client together with additional information.
- There are many implementations.



Source: Netcraft

Quiz

1. Perform a timing diagram of the two requests of Activity 2, including the establishment of the TCP connection.
2. Provide various examples of systems and services using HTTP in addition to the Web.
3. How many TCP connections are established when you visit www.firefox.com? And if you use version 1.0 of TCP?
4. Identify and describe the parts of a URL.
5. What is the advantage of using relative URLs?
6. Write the HTTP request without headers which will be generated when entering http://www.upct.es/ad/index.html. And when entering www.google.com
7. How does a browser know the type of content received from a server?
8. Explain what it means that HTTP does not store status information and why a "shopping cart” can not be implemented only with HTTP as described

Bibliography and extras

- Operation of the browser (also in Spanish): http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/
- The Apache Foundation, provides multiple free implementations of libraries and servers, not just HTTP servers: http://www.apache.org/