FEUP

Departamento de Engenharia Electrotécnica e de Computadores

# Services on the Web: servers and proxies


# Architectural components of the Web

- Web pages, which are hypermedia documents
  - "hyper" because they may contain selectable links to other documents
  - "media" because they can be composed by different data modalities
    - text, image, graphics, etc
- Web servers that store the Web pages
  - software that manages pages and grant clients access to those pages
- Web browsers, which are application programs that allow access to and presentation of Web pages
  - are clients to the Web servers, communicating through well-defined protocol, HTTP (Hyper Text Transfer Protocol)
  - normally designated as User Agents

# Architectural components of the Web (2)

- Web pages are represented using HyperText Markup Language (HTML)
  - text and embedded commands - tags - that provide indication to the Web browser on how to present the page
- each Web page is identified through a unique name
  - Uniform Resource Locator (URL) a special type of Uniform Resource Identifier (URI), normally adopting the HTTP scheme
    - formatted string identifying a resource with name, location, or any other characteristic
    - it needs to contain the complete location, i.e., the name of the server as well
      - unless the name of the server is implicitly known
      - http:// hostname [: port] / path [:parameters] [?query]

# Protocols

- HTTP is used for communication between UAs (normally Web browsers) and Web servers or intermediate entities
- main characteristics
  - text-based protocol
  - application-level, running on top of TCP/IP
  - request/response paradigm
  - stateless
    - each request or session is self-contained - the server does not keep history of previous sessions
  - bi-directional
    - the UA can also transfer data to the server

# Protocols

- main characteristics of HTTP (2)
  - capability negotiation
    - servers can indicate the capabilities they offer and the UA those that it can accept
  - caching in the client or intermediate entity
    - client can cache page and still interrogate the server to see if there have been updates since the cache was done
  - support intermediaries
    - entities that act as proxies

# HTTP GET request

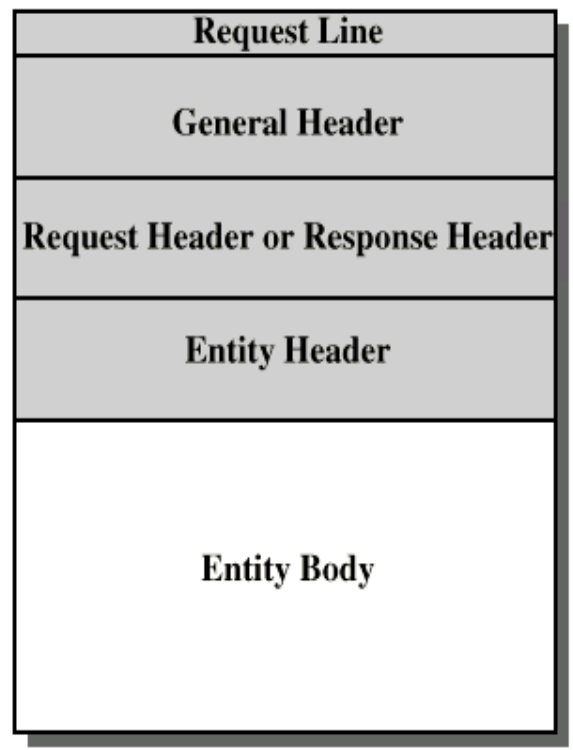
- the most usual form of HTTP communication is for a client to request some page/content to a server and the server to reply sending a copy of the requested data
  - GET command/method issued by the browser after the TCP connection has been established

    GET http://sigarra.up.pt/feup/web_page.inicial HTTP/1.1
- the server reply always starts with a 3-digit numeric code
  - if the code indicates that the method was successfully processed, it is followed by the requested data
  - if the code indicates failure or need for additional data from the client, no data follows

# HTTP messages

- Requests
  - Client to server
- Responses
  - Server to client
- Request line
- Response line
- General header
- Request header
- Response header
- Entity header
- Entity body

- the headers allow servers and clients to exchange metadata
  - concerning the entities themselves or describing the infomation that is going to be transacted



# Request message and methods

- Request line followed by
  - general, request header and entity headers followed by the entity body
- Request-Line
  - method <SP> Request_URL <SP> HTTP-Version <CRLF>
- Available methods:
  - Options
  - Get
  - Head
  - Post
  - Put
  - Patch
  - Copy
  - Move
  - Delete
  - Link
  - Unlink
  - Trace
  - Wrapped

# methods details

- OPTIONS
  - to allow a client to determine the options and/or requirements associated with a resource, or the capabilities of a server, without implying a resource action or initiating a resource retrieval
- GET
  - to retrieve information (in the form of an entity) identified by the Request-URI
- HEAD
  - identical to GET except that the server must not return a message-body in the response
  - used for obtaining metainformation about the entity implied by the request without transferring the entity-body itself (the resource)

# methods details (2)

- POST
  - used by the client to request that the server accept the entity enclosed in the request
- PUT
  - request to store the enclosed entity under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity should be considered as a modified version of the one residing on the server
- The fundamental difference between POST and PUT is reflected in the different meaning of the URI sent in the request
  - the supplied URI in a POST identifies the resource that will handle the enclosed entity (the resource/object being sent to the server)
  - the supplied URI in a PUT identifies the URI under which the enclosed entity should be stored

# methods details (3)

- DELETE
  - requests that the resource identified by the Request-URI be deleted by the server
- TRACE
  - used to invoke a remote, application-layer loop-back of the request message
  - allows the client to see what is being received at the other end of the request chain and use that data for testing or diagnostic information
- CONNECT
  - for use with a proxy that can dynamically switch to being a tunnel (e.g. SSL tunneling)

# Response message

- Status line followed by
  - one or more general, response and entity headers
  - optional entity body
- Status-Line
  - HTTP-Version <SP> Status-Code <SP> Reason-Phrase <CRLF>

# Server reply status codes

| code | meaning |
|---|---|
| 2xx | *success*: requested data follows in the body section of the reply |
| 200 | request fulfilled |
| 201 | *created*: the POST method issued by the client was successfully implemented; the URI of the created document is sent in the response line |
| 202 | *accepted but not yet processed* |
| 203 | *partial:* the data sent is not the definitive set |
| 204 | *no response*: there is not data to send back |
| 4xx, 5xx | *error*: in method issued by the client(4xx) or in the server (5xx) |
| 400 | *bad request:* wrong syntax or not possible to satisfy |
| 401 | *unauthorized*: the client should retry including the proper *Authorization* header |
| 402 | *payment required*: the client shoudl retry with a suitable *ChargeTo* header |
| 403 | *forbidden*: requested object is forbidden even with the *Authorization* header |
| 404 | *not found*: there is no object matching the URI provided in the request |

# Server reply status codes (2)

| code | meaning |
|---|---|
| 500 | *internal error*: unexpected condition in the server |
| 501 | *not implemented*: the server does not support the issued method |
| 502 | *overloaded*: the server is temporarily overloaded |
| 503 | *gateway timeout*: equivalent to 500 but in a third device that the server needs to access to fulfiill the request |
| 3xx | *redirection*: indicate further action to be taken by the client to fulfill the request |
| 301 | *moved*: the requested object was assigned a new URI; the response provides that URI |
| 302 | *found*: the requested object was assigned a new URI but the redirection may be altered |
| 304 | *not modified*: in response to a conditional GET, when the object being requested has not been modified since the date indicated in the field *If-Modified-Since* |

# General header fields

- Applicable to be used both in request and response messages
  - Cache control
  - Connection
  - Data
  - Forwarded
  - Keep alive
  - MIME version
  - Pragma
  - Upgrade

# Request Header Fields

- Accept
- Accept charset
- Accept encoding
- Accept language
- Authorization
- Expect
- From
- Host
- If-Match
- If-Modified-Since
- If-None-Match
- If-Range
- If-Unmodified-Since
- Max-Forwards
- Proxy authorization
- Range
- Referrer
- User-agent

# Response header fields

- ETag
  - entity tag for the requested variant, for caching purposes
- Location
  - location to where the client should be redirected
- Proxy authentication
  - authentication scheme and parameters that the proxy should provide
- Public
- Retry after
  - indicates how long the service is expected to be unavailable to the requesting client
- Server
  - information about the software used by the server to handle the request
- Vary
  - used to indicate whether a cache is permitted to use the response to reply to a subsequent request without revalidation.
- WWW-Authenticate
  - authentication scheme and parameters that the UA must provide to acces the requested URI

# Entity header fields

- Meta-information about an entity body or resource
  - Allow
  - Content encoding
  - Content language
  - Content length
  - Content MD5
  - Content range
  - Content type
  - Content version
  - Derived from
  - Expires
  - Last modified
  - Link
  - Title
  - Transfer encoding
  - URL header
  - Extension header

# Entity body

- Arbitrary sequence of octets
- HTTP can transfer any type of data including:
  - text
  - binary data
  - audio
  - images
  - video
- data is interpreted based on value of header fields
  - Content encoding, content type, transfer encoding

# Persistent connections

- first version of HTTP uses one TCP connection for each transfer
  - client opens TCP connection, sends request; server sends reply and closes the connection
    - client reads until EOF and then closes the connection
- HTTP 1.1 uses persistent connections as default
  - the TCP connection stays open and can be used by the client to place mltiple requests
  - both the client and the server may decide to close it, informing the other party of that intention
  - it alows pipelining requests
    - sending multiple requests without waiting for the response
    - can be useful when retrieving multiple images of a given Web page
  - it requires signalling the begining and end of each item sent over the connection
    - the length of the item is sent before the item

# Persistent connections (2)

- sending the length before the item may not aways be possible
  - some Web pages are generated on-demand
    - dynamic Web pages generation, for example, using Computer Gateway Interface (CGI)
    - when a request arrives corresponding to a CGI-generated Web page, the server runs the CGI program and sends the result to the client
    - the server does not know the length of the item a-priori
    - in this case the server closes the connection after sending the data
- the length is sent using the field Content-Length in the entity header
- if the length is not known, the server sends a Connection field in the general header

# Negotiation

- some fields in the headers allow servers and clients to negotiate capabilities
  - characteristics of the connection
  - representation of content (e.g., compression schemes used)
  - type of content (e.g., language used)
  - control (e.g., time a page remains valid)
- server-driven negotiation
  - based on the preferences expressed by the client request, the server selects the best representation from the available ones
- UA-driven
  - 2-step processs to perform the selection of the best representation
    - the UA sends a request asking the server what is available
    - the server returns a list of possibilities
    - the client selects one and sends a second request for the selected possibility

# server-driven negotiation

- the UA uses the field Accept in the Request header to specify which media, formats, representations are acceptable to it
  - Accept: text/html, text/plain; q=0.5, text/x-dvi; q=0.8
    - *the browser accepts text/html but if that does not exist, the 2nd preference is x-dvi (with a weight of 0.8) and the 3rd preference is text/plain (with weight of 0.5)*
    - *text/html does not have a weight associated which means a maximum weight of 1*
    - *weight of 0 means that the type is unacceptable*
  - there are different Accept fields
    - Accept (list of MIME types of the media that the UA can process)
    - Accept-Encoding
    - Accept-Charset
    - Accept-Language
- can use also the header User Agent
  - to specify characteristis of the UA that issued the request

# UA-driven negotiation

- selection of the best representation for a response is performed by the UA after receiving an initial response from the origin server
  - based on a list of the available representations of the response included within the header fields or entity-body of the initial response
  - each representation is identified by its own URI
  - may be performed automatically, if the UA has the  is capability or manually by the user
- has the disadvantage of being a 2-step process
  - the first request is used to obtain information about the options available in the server

# server-driven negotiation (2)

- There are disadvantages of implementing server-driven negotiation
  - the server cannot accurately determine what might be "best" for a given user
    - it would need complete knowledge of both the capabilities of the user agent and the intended use for the response
  - it requires the UA to describe its capabilities in every request
    - might be pointless and thus inefficient, given that only a small percentage of resources/responses will have multiple representations
    - can be a potential violation of the user's privacy
  - it complicates the implementation of the server and the algorithms for generating responses to a request
  - it may limit a public cache's ability to use the same response for multiple user's request

# Conditional Requests

- a client can send a request to a server for a page in a conditional mode
  - the server will only send it the requested page if the condition is met
  - allows to optimize retrievals, avoiding unecessary transfers
  - the condition is specified in the header of the method

| conditional header | parameter and description |
|---|---|
| *If-Modified-Since* | *date time value*: the operation should only be performed if the requested resource was modified since the specifie date time |
| *If-Unmodified-Since* | *date time value*: the operation should only be performed if the requested resource was not modified since the specifie date time |
| *If-Match* | *ETag value*: the operation should only be performed if the requested resource's ETag matches the one provided |
| *If-None-Match* | *ETag value*: the operation should only be performed if the requested resource's ETag does not match the one provided; the specified ETag can be * to indicate that the operation should only be performed if the resource does not exist |

An ETag, Entitity Tag, is an identifier assigned by the server to a resource in a given URL; when the resource is modified, so is its ETag; can be seen as a fingerprint of the resources

# Proxy servers

- important components of the Web
  - allow to decrease latency and reduce the load of servers
- two types of proxies
  - nontransparent
    - are visible to the user who needs to configure his browser to contact the proxy instead of contacting directly the source
    - normally through the registered port 3128
  - transparent
    - users do not need to know the proxy is running
    - the proxy examines all TCP connections and intercepts traffic to port 80
- all proxies cache data and thus may directly answer the client's request

# HTTP support to Proxy servers

- HTTP specifies
  - how proxies should handle requests and interpret headers
  - how clients should negotiate with proxies
  - how proxies should negotiate with servers
- reserves headers for use with proxies
  - to allow the proxy to authenticate itself to the server
  - to allow the proxy to include its identification in the reply
    - the ultimate recipient will thus know all the intermediaries
- allows to control the way proxies handle Web pages
  - e.g., Max-Forwards to limit the number of intermediate proxies

# Caching

- ultimate goal is to improve performance
  - reducing latency and alleviating servers' load
- major aspect to consider
  - how long should a page be cached
- HTTP allows for the server to control how caching is performed
  - in the response it can include caching details
    - whether the page can be cached
    - how long a page can be cached
      - the time at which the cache expires
    - the audience to which the cached page can be disseminated
    - limitatios to the transformations that can be done in the cached page
- HTTP provides the tools for clients to force revalidation of a page
  - using a header in the GET method indicating that the age of the page cannot be greater than zero

# HTTP security

- there are essentially 2 main aspects related with security on the Web
  - the integrity and confidentiality of the data being transferred
  - the authenticity of the web site offering items
- the base protocol does not provide security mechanisms
  - it relies on the use of other protocols and tools
    - encryption can be used to ensure data confidentiality
    - certificates can be used to ensure authenticity of Web sites
    - Secure Socket Layer (SSL) protocol
      - HTTP over SSL - HTTPS

# Emergent standards

- HTML 5
  - HTML was invented to structure and package documents, not Web applications
  - HTML 5 addresses the needs of the modern Web, with extensive support (APIs) for interaction between content and the local computer
    - draw arbitrary graphics
    - find your position on the globe
    - cache code and data
    - offload compute-intensive tasks to keep the interactive portion of the browser responsive
  - HTML 5 formalizes ad hoc techniques currently used to structure content
    - among others, Web page constructs such as , , , and <ul class="nav"> are replaced with the <header>, <footer>, <article>, and <nav> tags
  - Although not yet an approved standard, most of the leading Web browsers already support its APIs (e.g., Chrome, Firefox, Safari, and Opera)

# References


- Comer, Douglas E., "Internetworking With TCP/IP Vol I: Principles, Protocols, and Architectures". 5th Ed. Prentice Hall, 2006
- RFCs 1945, 2616
- http://www.w3.org/Protocols/Specs.html
- http://www8.org/w8-papers/5c-protocols/key/key.html