HTTP - HyperText Transfer Protocol

WASA: Web and Software Architecture

Prof. Emanuele Panizzi

HTTP - HyperText Transfer Protocol

- Application layer protocol in the Internet protocol
- Invented by **Tim Berners-Lee** @ CERN 1989-1991
- HTTP/3 published in 2022
- secure variant named HTTPS

Version	Year introduced	Current status
HTTP/0.9	1991	Obsolete
HTTP/1.0	1996	Obsolete
HTTP/1.1	1997	Standard
HTTP/2	2015	Standard
HTTP/3	2022	Standard

Client/Server

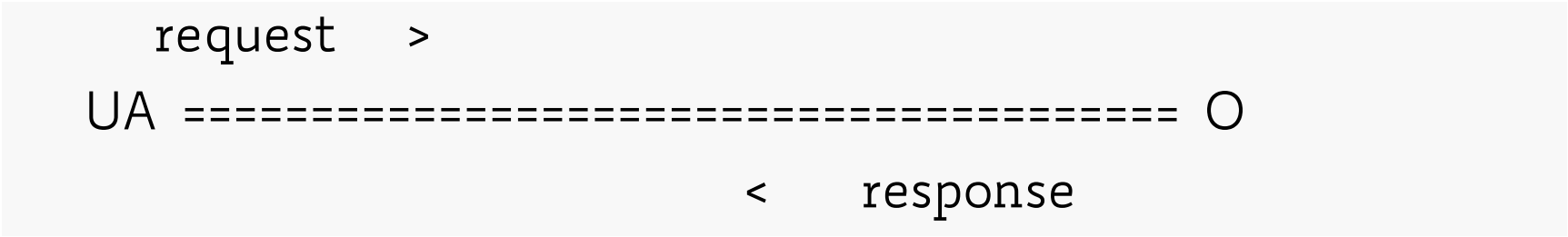
Client

- **User Agent** (UA): any of the various client programs that initiate a request (e.g. web browser, mobile app, spider, household appliances,...)

Server

- **Origin Server** (O): a program that can originate authoritative responses for a given resource (e.g. web site, traffic camera, office machines, video-on-demand platforms,...)

Request/response



Example – request

- Request line (HTTP-method URI protocol-version)
- Request header fields
- (optional) message body

```
GET /hello.txt HTTP/1.1
User-Agent: curl/7.64.1
Host: www.example.com
Accept-Language: en, it
```

Example – response

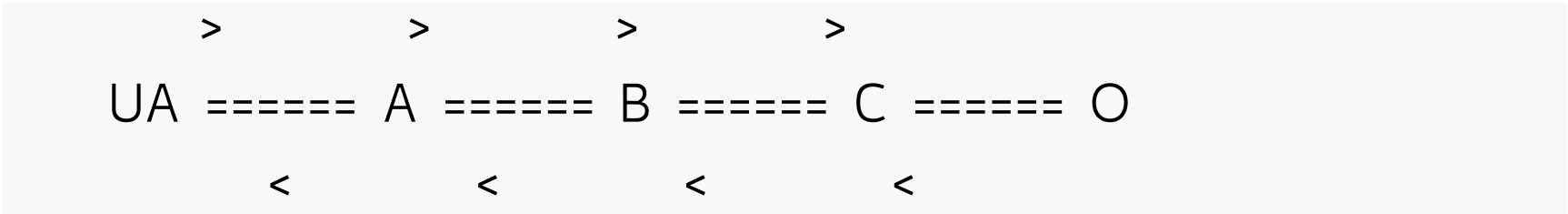
- completion status (about the request)
- (may contain) content

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 51
Vary: Accept-Encoding
Content-Type: text/plain

Hello World! My content includes a trailing CRLF.
```

Intermediaries

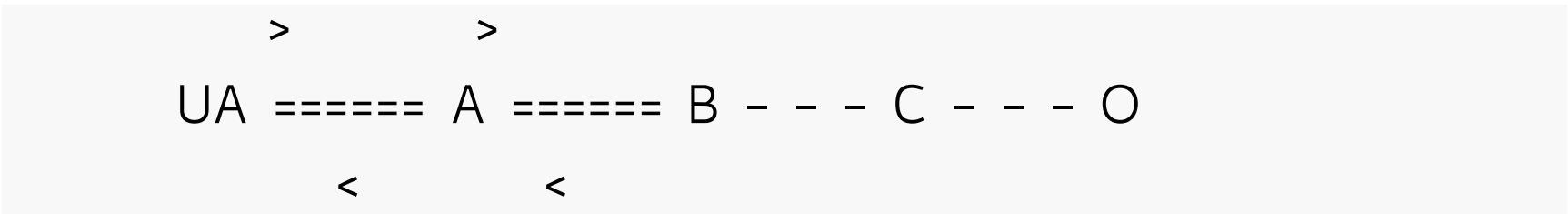
chain of connections



E.g. Proxies, gateways, tunnels.

Caches

A store of previous response messages.



- O must declare a response as cacheable
- proxies can cache responses
- tunnels cannot cache responses

HTTP Methods

HTTP Methods

- GET
- HEAD
- POST
- PUT
- DELETE
- CONNECT
- OPTIONS
- TRACE
- PATCH

Methods properties

SAFE

A method that has no side effect on the resource, i.e. it is 'read-only'. It can however change the state of the server in other ways (e.g. logs) GET, HEAD, OPTIONS and TRACE are safe.

IDEMPOTENT

Multiple identical requests with that method have the same effect as a single such request. PUT and DELETE are idempotent.

CACHEABLE

Methods that can allow a cache to store and use a response. GET, HEAD, and POST under some conditions, are cacheable.

PUT

- Create a new resource, specifying it in the request
- Replace a resource (when the URI exists, overwrite it with the representation in the PUT payload)

```
PUT /course-descriptions/web-and-software-architecture
```

- **IDEMPOTENT:** any successive identical PUT request does not modify the resource
- Neither safe, nor cacheable

GET

- Request a representation of the state of a resource

```
GET /course-descriptions/web-and-software-architecture
```

- **SAFE**: no changes to the resource
- **CACHEABLE**: response can be cached by an intermediary, and reused without asking for it to O; conditions (e.g. expire date, etc.) can be specified.
- Not idempotent

POST

- Create or modify a subordinate of the resource indicated in the URI. The URI identifies the resource that will handle the request.

```
POST /announcements/
POST /announcements/{id}/comments/
POST /users/{id}/email
```

- The action might not result in a new resource.
- Response can be cacheable if specified.
- Not safe.
- Not idempotent! E.g. the first example above adds many new identical announcements if requested many times.

DELETE

- Request that the origin server removes the association between the target resource and its current functionality

```
DELETE /courses/web-and-software-architecture
```

- **IDEMPOTENT**: deleting an already-deleted resource does not produce any new effect
- It is neither safe nor cacheable

Other methods

Method	Description
HEAD	Same as GET, but do not transfer the response content.
CONNECT	Establish a tunnel to the server identified by the target resource.
OPTIONS	Describe the communication options for the target resource.
TRACE	Perform a message loop-back test along the path to the target resource.

Properties of request methods

Request method	RFC	Request has payload body	Response has payload body	Safe	Idempotent	Cacheable
GET	RFC 7231	Optional	Yes	Yes	Yes	Yes
HEAD	RFC 7231	Optional	No	Yes	Yes	Yes
POST	RFC 7231	Yes	Yes	No	No	Yes
PUT	RFC 7231	Yes	Yes	No	Yes	No
DELETE	RFC 7231	Optional	Yes	No	Yes	No
CONNECT	RFC 7231	Optional	Yes	No	No	No
OPTIONS	RFC 7231	Optional	Yes	Yes	Yes	No
TRACE	RFC 7231	No	Yes	Yes	Yes	No
PATCH	RFC 5789	Yes	Yes	No	No	No

Response status codes

e.g. HTTP/1.1 200 OK

- Describes the result of the request and the semantics of the response.
  - whether the request is successful
  - what content is enclosed (if any)
- Three-digit number in the range 100–599

Status code classes

status code	description
1xx	(Informational): The request was received, continuing process
2xx	(Successful): The request was successfully received, understood, and accepted
3xx	(Redirection): Further action needs to be taken in order to complete the request
4xx	(Client Error): The request contains bad syntax or cannot be fulfilled
5xx	(Server Error): The server failed to fulfill an apparently valid request

Common status codes - 2xx success

200 OK In a GET request, the response will contain an entity corresponding to the requested resource; in a POST request, the response will contain an entity describing or containing the result of the action

201 Created The request has been fulfilled, resulting in the creation of a new resource

204 No Content The server successfully processed the request, and is not returning any content

Common status codes - 3xx redirection

301 Moved Permanently This and all future requests should be directed to the given URI
302 Found (Previously "Moved temporarily") Look at another URL

Common status codes - 4xx client errors

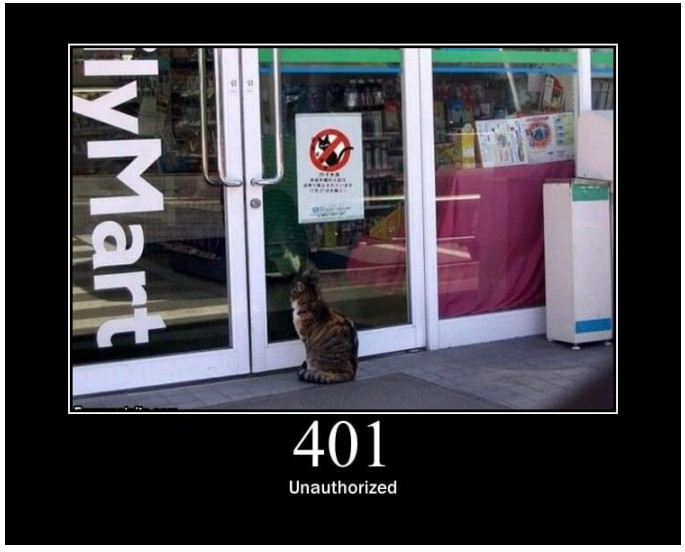
400 Bad Request Apparent client error

401 Unauthorized Authentication is required

403 Forbidden The request contained valid data and was understood by the server, but the action is prohibited

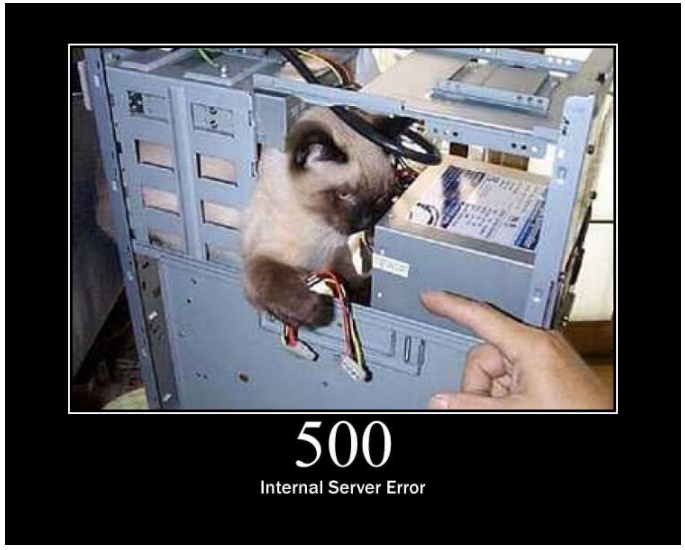
404 Not Found Resource could not be found but may be available in the future

405 Method Not Allowed The request method is not supported; e.g., a PUT request on a read-only resource.



Common status codes - 5xx server errors

500 Internal Server Error
Unexpected condition encountered
501 Not Implemented
Unrecognized request method, or the server lacks the ability to fulfil the request
502 Bad Gateway A gateway or proxy received an invalid response from the upstream server
503 Service Unavailable Server overloaded or down for maintenance (temporary)
504 Gateway Timeout The server did not receive a timely response from the upstream server



Links

- https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- https://en.wikipedia.org/wiki/List_of_HTTP_header_fields
- https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
- https://www.rfc-editor.org/rfc/rfc7231
- https://www.rfc-editor.org/rfc/rfc9110.html
- https://http.cat