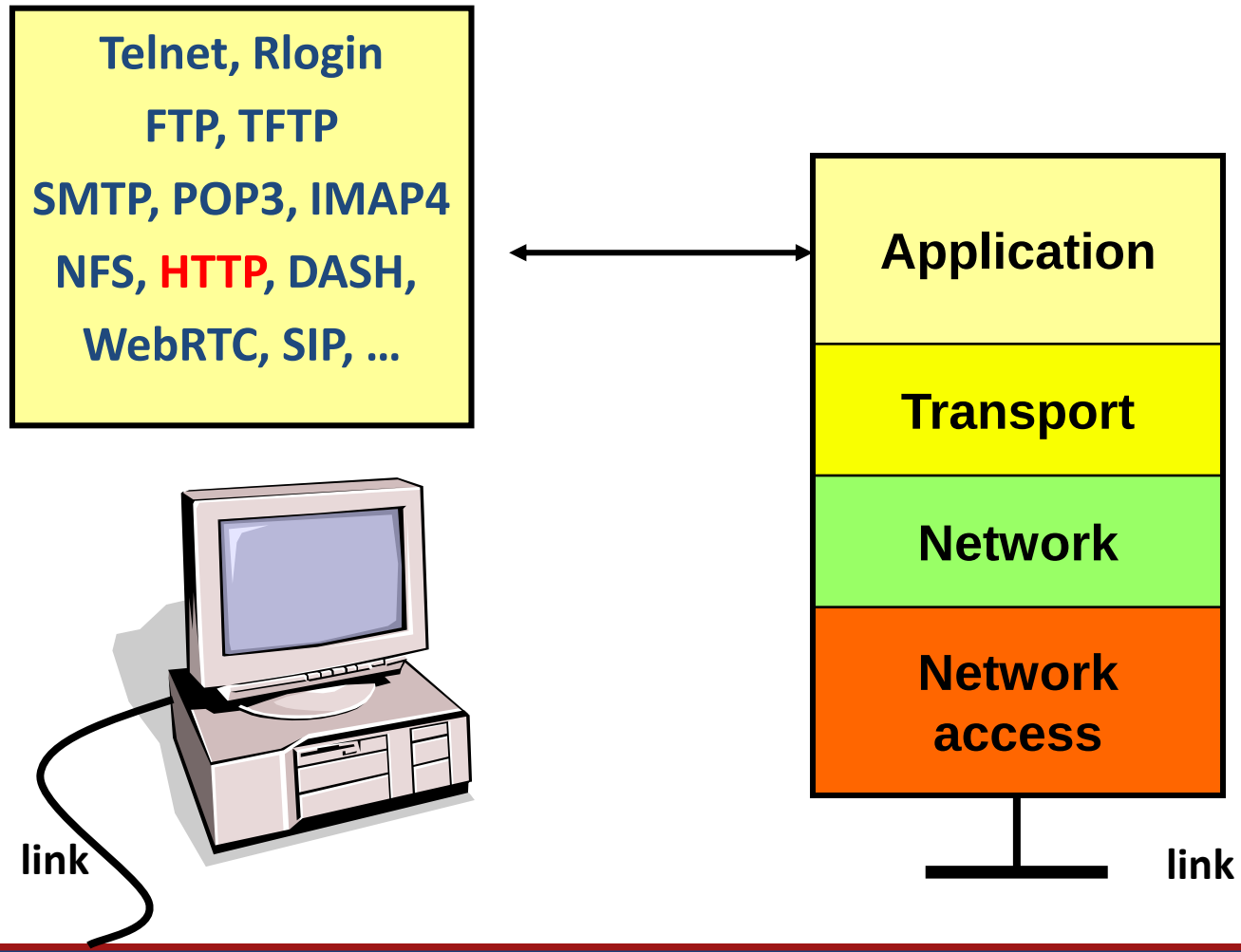
UNIVERSITY OF CAGLIARI

DIEE - Department of Electrical and Electronic Engineering

Infrastrutture ed Applicazioni Avanzate nell'Internet

Application protocols: HTTP, WebSocket and DASH

Application protocols

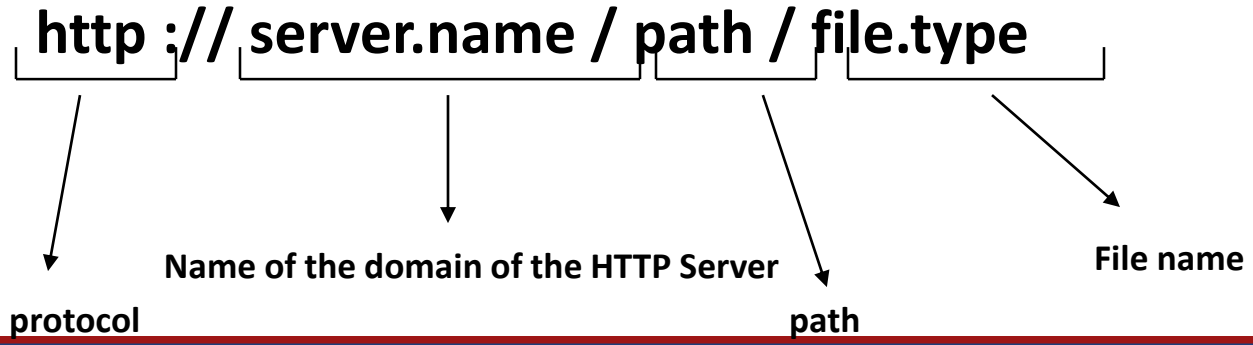


HyperText Transfer Protocol (HTTP) (1)

- It is the standard protocol to access to web servers
  - It is defined by the W3C (not an SDO)
- It has been defined as an I/O protocol for hypertextual environments
- It is a protocol without statuses -> stateless
  - But cookies and hidden variables can make it stateful
- Client and server communicate through TCP using port 80 (default)
- The client is a web browser
- The language for the description of the documents is the HyperText Markup Language (HTML)
  - Now version 5.2
- The protocol makes use of human-readable commands

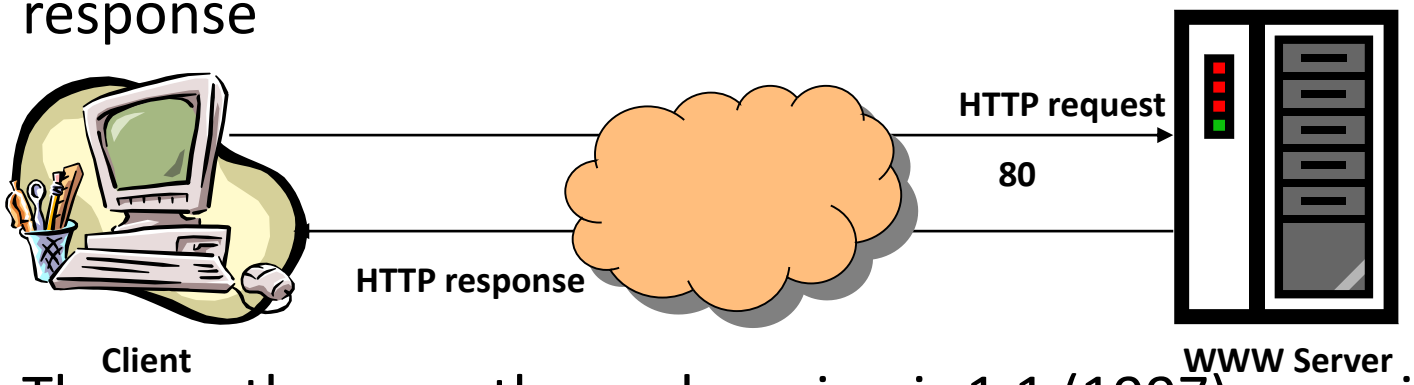
HTTP (2)

- URI (*Uniform Resource Indentifier*)
  - Generic term to refer to a method to identify a resource
  - The URL is a specific type of URI
- Each page is associated with a URL (Uniform Resource Locator), for instance:
  - http://diee.unica.it/data/soluzioni.html



HTTP: basic features

- It relies on a client/server interaction with request and response



- The mostly currently used version is 1.1 (1997), even if HTTP/2 and HTTP/3 exist
  - Persistent connections
    - One single TCP connection may be used for more than one request
  - Partial transfer of documents
  - It allows for the compression of contents
    - Files can be transferred in a compressed status
  - It allows for virtual hosts
  - It manages the authentication processes

HTTP transactions

- The browser starts a TCP connection and sends the request
- The server sends the requested file, an error message or a message saying the request has been processed
  - Then the server typically closes the connection

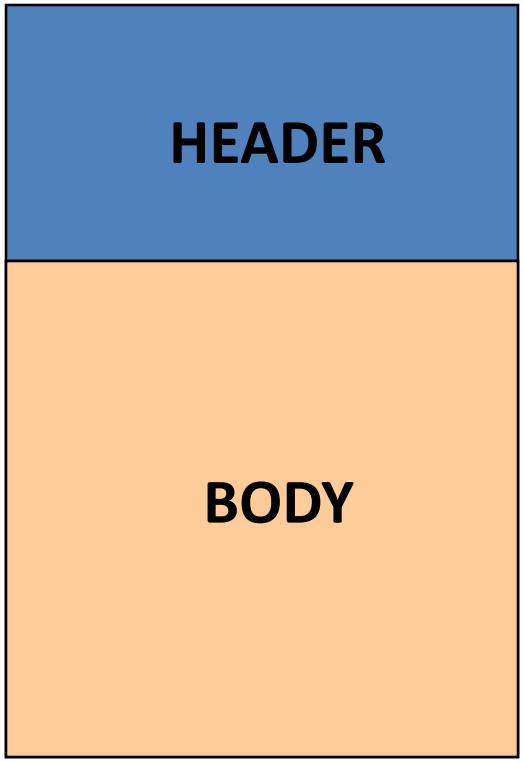
Generic request

<METHOD> <URI>
"HTTP/1.1"<CRLF>
<Header>: <Value>
…
<Header>: <Value>
<CRLF>
<BODY>

Generic response

HTTP/1.1 <STATUS_CODE>
<REASON><CRLF>
<Header>: <Value>
…
<Header>: <Value>
<CRLF>
<BODY>

HTTP: requests (1)



Possible requests

- GET (default)
- HEAD
- POST
- PUT
- DELETE
- TRACE
- CONNECT
- OPTIONS

HTTP: requests (2)

- **GET:** to get a resource from the server who then sends it to the client:
  - GET <URI> HTTP/1.1
- **HEAD:** as get but used to obtain only the header of the response but not the body
- **POST /PUT:** to send data to the server
- **DELETE:** the client asks the server to delete a resource
- **TRACE:** the server echoes the received request to see if proxies have changed the request
- **OPTIONS:** to know the options supported by the server for a given URL
- **CONNECT:** used by the client when it needs a tunnel of a TCP connection to be created by a proxy towards a given external server. This is used when all the communications from the clients should pass through a given proxy

HTTP: headers in HTTP messages

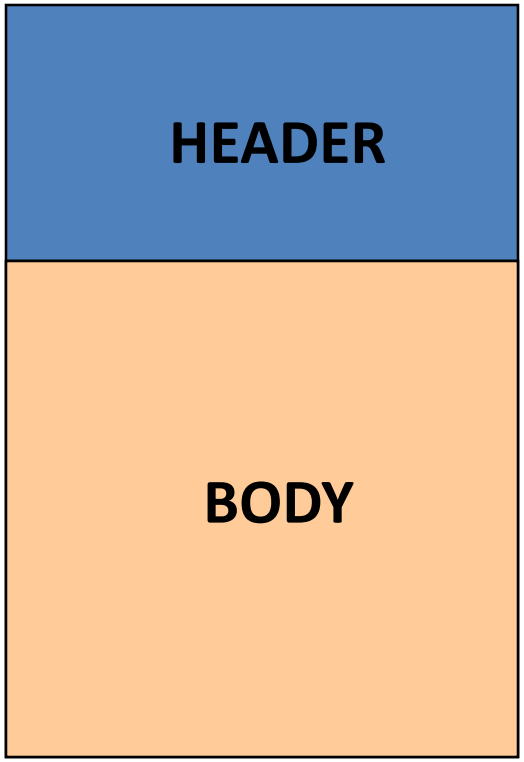
e.g. -> `Accept: text/html`

- FROM (e-mail address of the requestor)
- USER-AGENT (name of the client program)
- ACCEPT (types of files accepted in the response)
- ACCEPT-ENCODING (methods of compression accepted, e.g., gzip)
- ACCEPT-LANGUAGE (accepted languages)
- REFERER (address of the previous page)
- AUTHORIZATION (credentials for HTTP authentication)
- IF-MODIFIED-SINCE (send the document only if modified since)
- CONTENT LENGHT (lengths in bytes of the body of the message)
- CONNECTION (used to send the Keep Alive option to keep the connection open
- HOST (address of the host to which the request is addressed)
- COOKIE (to send information about the previously set cookie)

HTTP: *cookie*

- These are short amount of data that the server asks the client to save and that can be retrieved later on
- The client can refuse to save cookies and to remove previously saved cookies
- Two headers are used: Cookie and Set-Cookie
  - *Set-cookie* to request to save the cookie in the local memory
  - The browser sends back the cookie with the header *Cookie*
- Types:
  - Session cookie: used only for the current session
  - Permanent cookie: lasts for a given specified interval of time
  - Secure cookie: sent only over secure connections
- Structure
  - Name
  - Value
  - Attributes
    - path
    - secure
    - comment
    - max-age
    - version
- Uses: session management, personalization, tracking

HTTP: responses



The first line provides:

- HTTP version
- Status code
- Code description

Format of the code status

1xx information
2xx success
3xx redirection
4xx client error
5xx server error

HTTP: status codes, examples

- 200: ok
- 201: created
- 202: accepted
- 204: no content
- 300: multiple choices
- 301: moved permanently
- 302: found
- 304: not modified
- 400: bad request
- 401: unauthorized
- 403: forbidden
- 404: not found
- 500: internal server error
- 501: not implemented
- 502: bad gateway
- 503: service unvaliable

HTTP: header in the responses

- SERVER (name and version of the web program)
- DATE (when the response has been created)
- LAST MODIFIED (date of last change)
- EXPIRES (validity of the send document)
- CONTENT-TYPE (MIME type of the data)
- CONTENT LENGHT (length of the sent message in bytes)
- WWW AUTHENTICATE (the authentication of the user is required)
- SET COOKIE (as already said)

HTTP: example

http://www.somehost.com/abc/saluti.html

```
GET /abc/saluti.html HTTP/1.1
From: someuser@jmarshall.com
Accept: text/html
Accept: text/plain
Accept: image/gif
User-Agent: Netscape 1.1N PPC
[linee vuote]
```

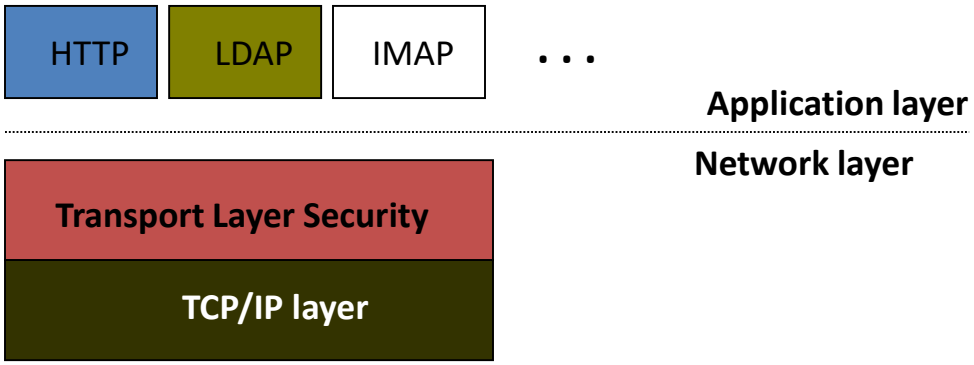
```
HTTP/1.1 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/html
Content-Length: 1354

<html>
<body>
<h1>Ciao a tutti.</h1>
(more file contents)
...
</body>
</html>
```

HTTP: general aspects

- If the HTTP response does not contain the *Content Lenght* header, the message is intended to be finished and the TCP connection is closed (unless keep-alive header was sent by the client)
- The response can contain MIME multipart objects
  - It allows for sending contents coded in different formats other than the default ASCII one

Hypertext Transfer Protocol Secure (HTTPS)

- It is the HTTP that makes use of the Transport Layer Security (TLS) over port 443. It assures
  - Privacy, authenticity and integrity
  - It protects against man-in-the-middle attacks
  - Historically used for financial transactions, now used to preserve privacy in (almost) any user communications



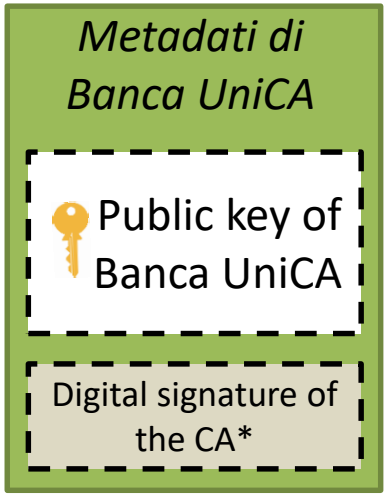
HTTPS: Authentication

- HTTPS follows the same syntax of HTTP, however it signals the browser to use an additional encryption layer (even if only one side is authenticated)
  - The bank server, for instance
- Everything on top of the TLS layers is encrypted (included all the HTTP messages
  - However, host address and port number cannot be obfuscated (included length of the session)
- Authentication: it relies on certificate authorities (CA)
- The communication is secure if
  - The user trusts the browser functionalities (and pre-installed CA cert.)
  - The user trusts the certification authority and the website provides a relevant valid certificate (signed by a trust authority)
  - The certificate is valid for the website we intend to visit
  - The user trusts the TLS layer that is being used

HTTPS: Authentication

- The process works with the asymmetric cryptography
- The CA public key is know to the Internet users (browsers)
- When the browser connects to the website, it receives a *certificate*
  - The user deciphers the certificate with the CA public key -> it allows the user to trust the content of the certificate
  - The user knows then that the contained public key of the website is valid
  - This is used to generate a private session key which is used during the session

  * Content + public key encrypted with the CA private key



Certificate of Banca UniCA

- A man-in-the-middle can send a different certificate
  - It cannot be certified by a CA with a different public key
  - It can put the same public key
    - Can decode only content with the private key of the real website, which is however unknown by the hacker
- The certificates are typically encoded with the X.509 standard
- *Identify the CA certificates and those of common websites*

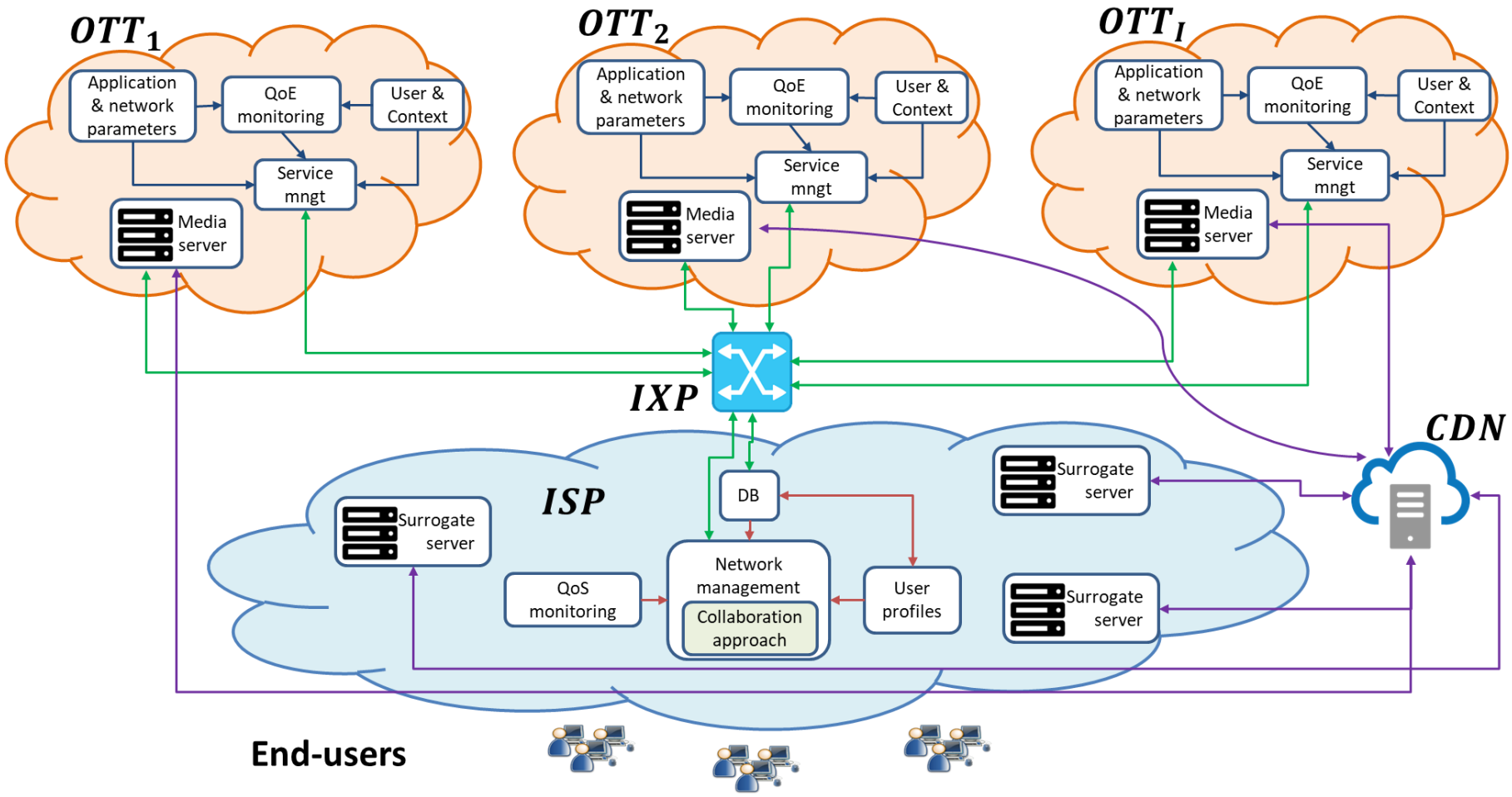
HTTP: Web cache

- *System in forward position (recipient or client side)*
  - It is a cache outside the web server's network (for instance on the client computer, in an ISP or within a corporate network). A web proxy sitting between the client and the server can evaluate HTTP headers and choose whether to store web content.
- *System in reverse position (content provider or web-server side)*
  - It is a cache in front of one or more web servers and web applications, accelerating requests from the Internet
  - A content delivery network (CDN) belongs to this category
  - A search engine may also cache a website (e.g., Google)

HTTP: proxy server

- *Open proxy*
  - An open proxy forwarding requests from and to anywhere on the Internet
  - If *anonymous,* it does not reveal the address of the client. Otherwise is *transparent*
- *Reverse proxy* (or *surrogate*)
  - It appears to be the origin server
  - These are installed in the neighbors of the original servers and serve only a restricted number of origin servers. Advantages
    - Encryption / SSL acceleration
    - Load balancing
    - Caching content
    - Compression
- A NAT (Network Address Translation) server is a layer-3/4 proxy

HTTP: the use of surrogate servers



HTTP: WebSocket

- Full-duplex communication over TCP (same ports of HTTP)
- For compatibility benefits, WebSocket uses the HTTP Upgrade header to change

```
GET /dispense HTTP/1.1
Host: tlc.diee.unica.it
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
Origin: http://unica.it
```

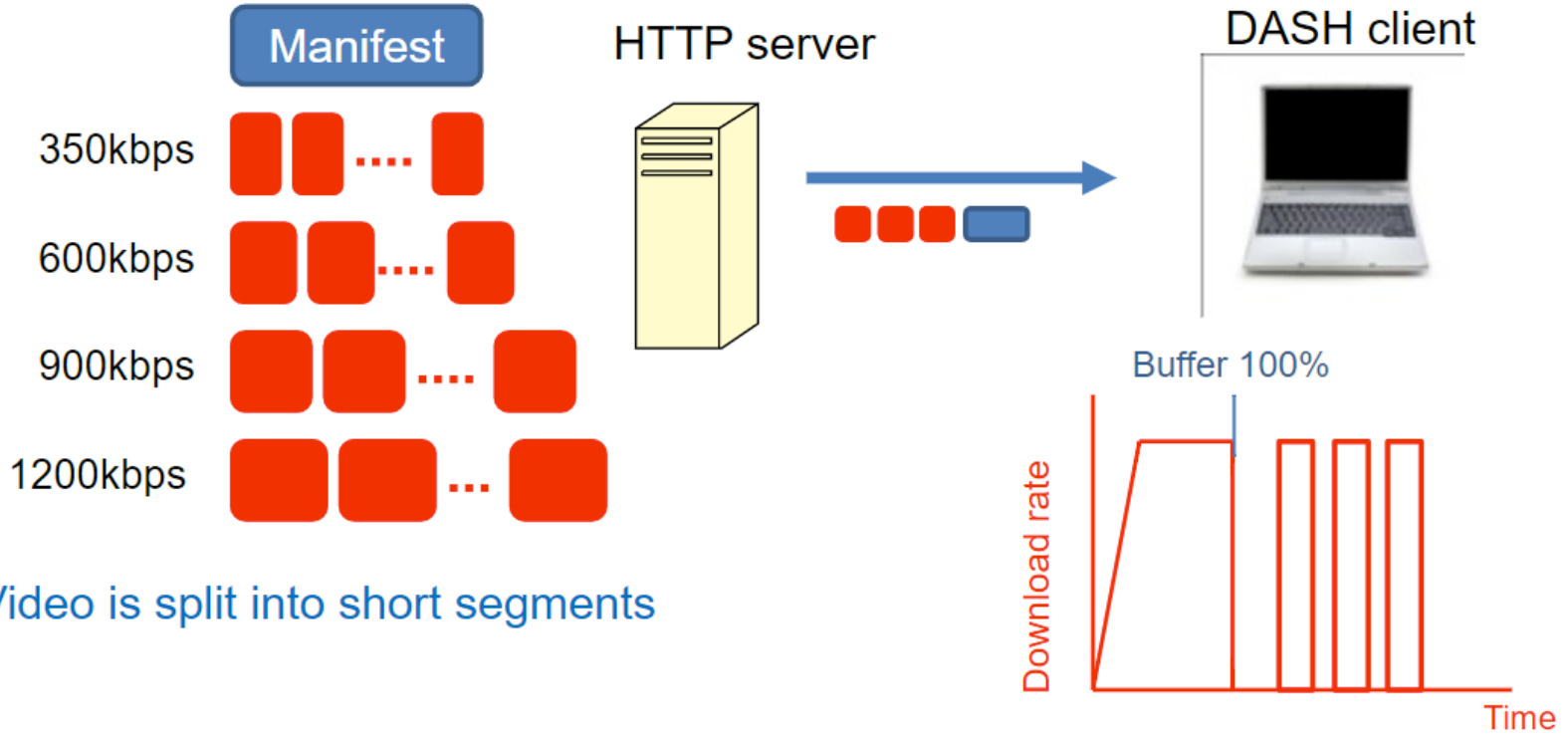
```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

- Once switched, the communications do not follow HTTP and it is binary
- It facilitates real-time data transfer from and to the server (ws:// or wss://)

HTTP: hands on activity

- Open the browser and connect to qoenet-itn.eu and capture the traffic with Wireshark
- Identify
  - The used HTTP methods and responses
  - The IP address of the server
  - Your IP address
  - The used HTTP headers and their meaning
- Try to find out another unsecure website

HTTP video streaming



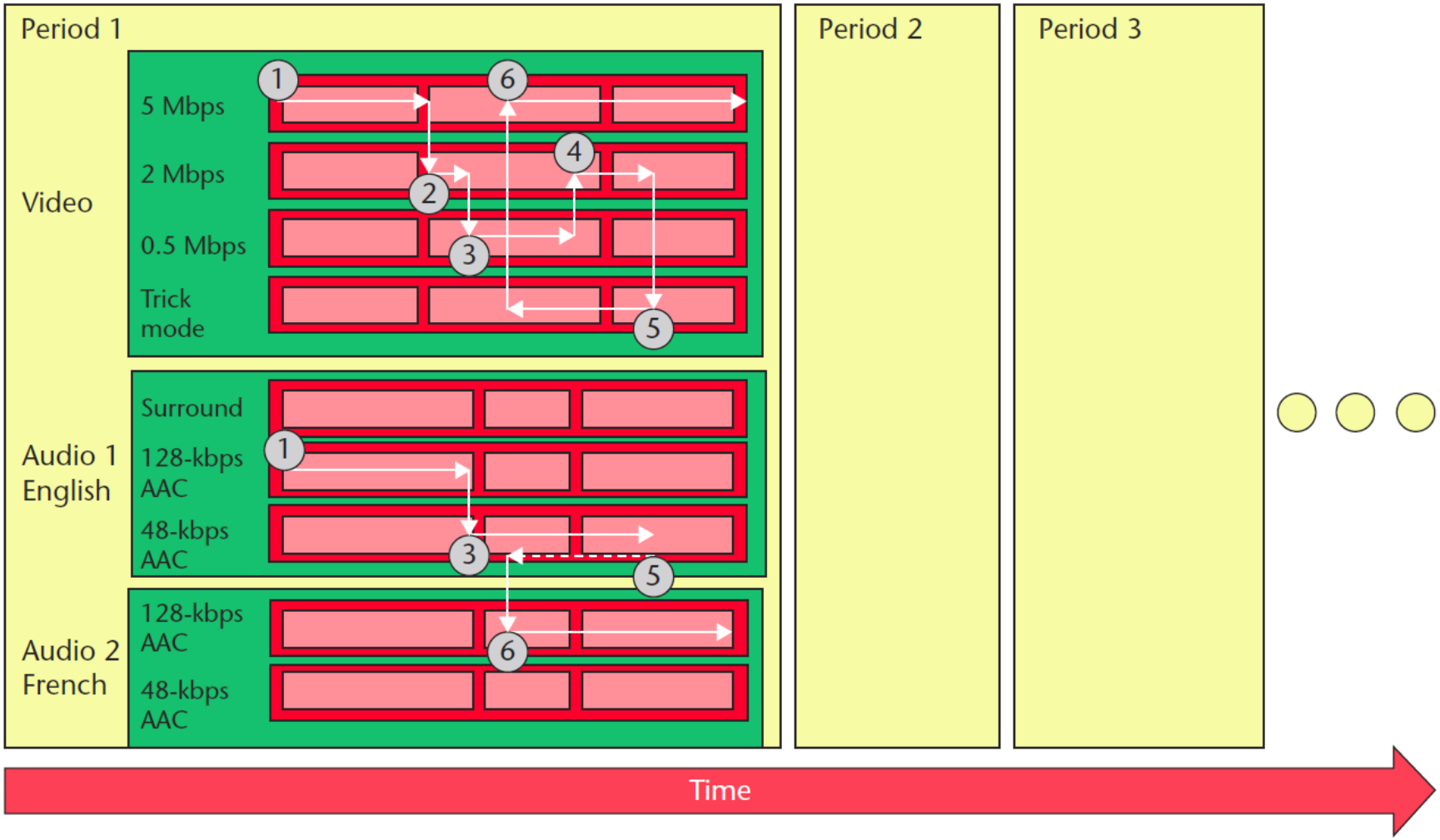
- Video over HTTP over TCP

MPEG-DASH -> Anthony Vetro, «The MPEG-DASH Standard for Multimedia Streaming Over the Internet," IEEE Multimedia, 2011

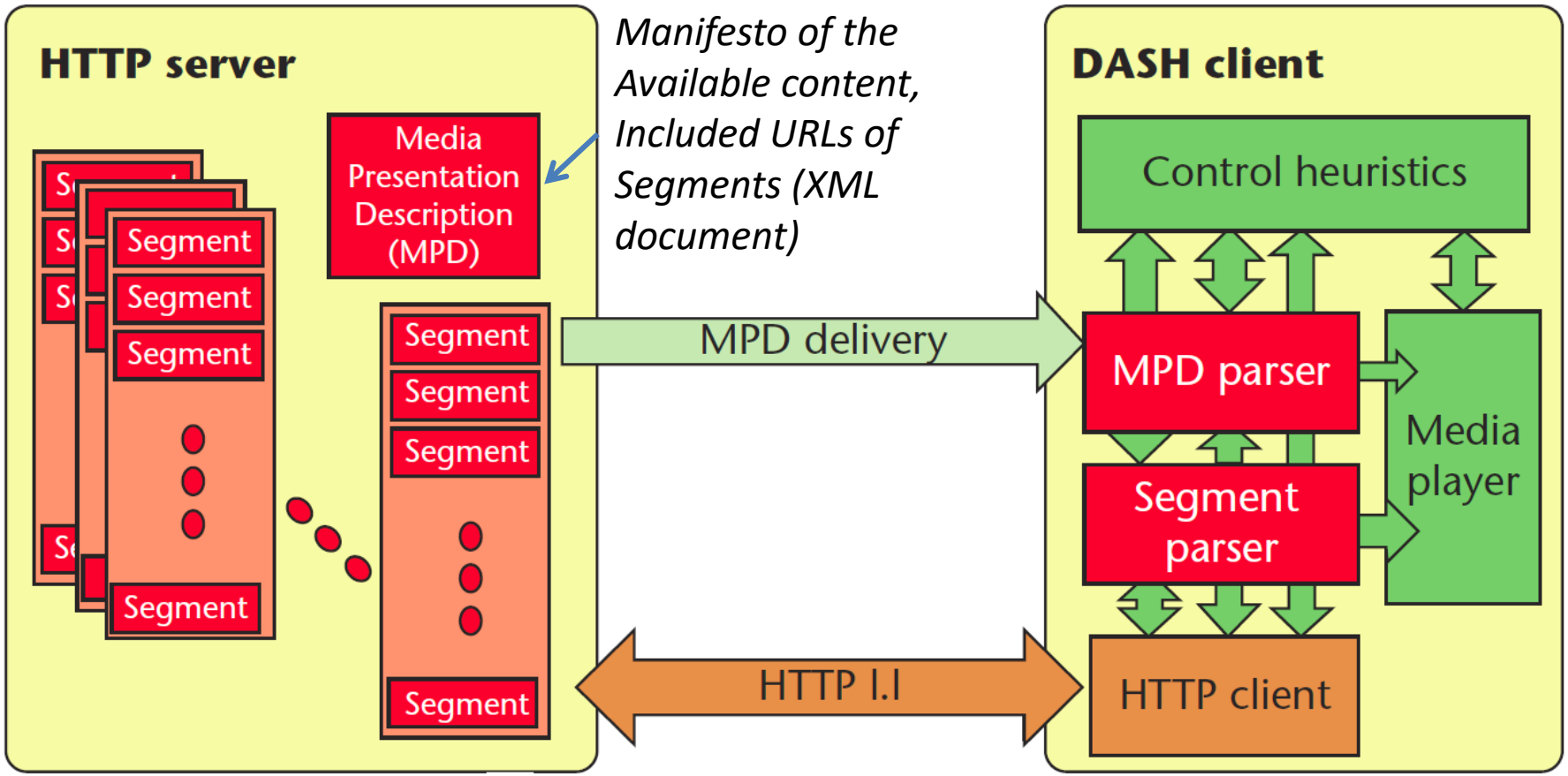
MPEG-DASH standard

- Different proprietary systems available on the market
  - Need for a standard!
  - Fostering the development of products
- History: Real-time Transport Protocol (RTP)
  - For managed IP nets, over UDP
  - Current situation: CDN over public IP (complex)
    - Network evolved towards HTTP traffic (caching included)
    - HTTP server: commodity, supports millions of users
      - Stateless
    - Security well managed
- Many use HTTP streaming: Apple, Adobe, Microsoft
  - Call for proposal in 2009 (MPEG, + 3GPP)

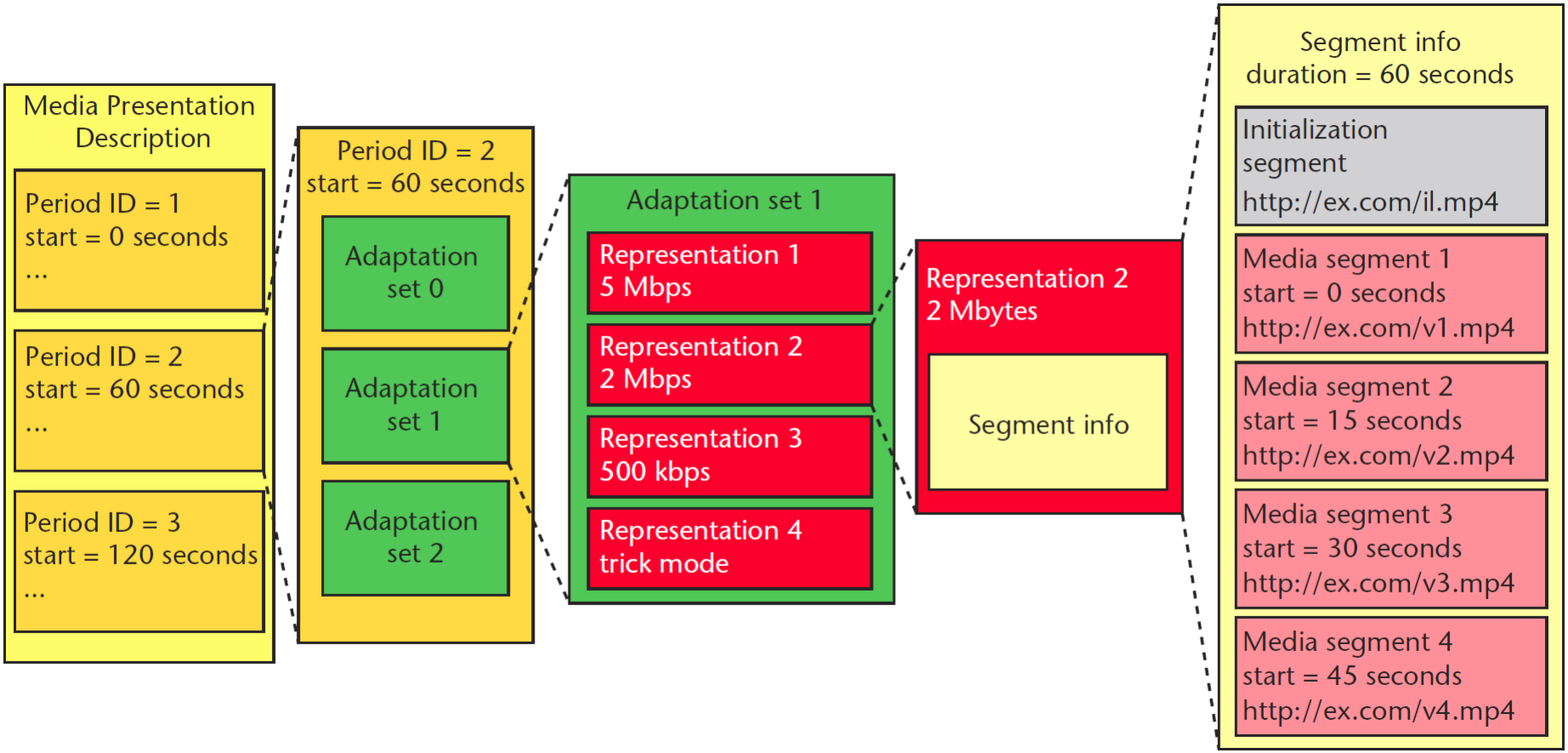
Dynamic Adaptive streaming: an example



Scope of the MPEG-DASH standard



Scope of the MPEG-DASH standard



MPEG-DASH: additional features

- Switching and selectable streams
- Ad insertion
- Compact manifest
- Fragment manifest
- Multiple base URLs
- A flexible set of descriptors
- Quality metrics for reporting the session experience

- Osserviamo alcune statistiche su youtube:
  - «stats for nerds»

Questions

- Which HTTP methods do you know?
- What is it meaning of «HTTP is human-readable»?
- Which are two HTTP headers you know?
- What is the MPD file used for?
- What is a certificate authority?
- Is HTTP stateful? What is the meaning?
- Which types of web caching systems do you know?
- What is a virtual host?
- What is the purpose of «keep-alive» header?
- What is the purpose of the DASH standard?

End of Application protocols: HTTP, WebSocket and DASH