HTTP Request Lifecycle

Western Greece Software Development Meetup, October 2017
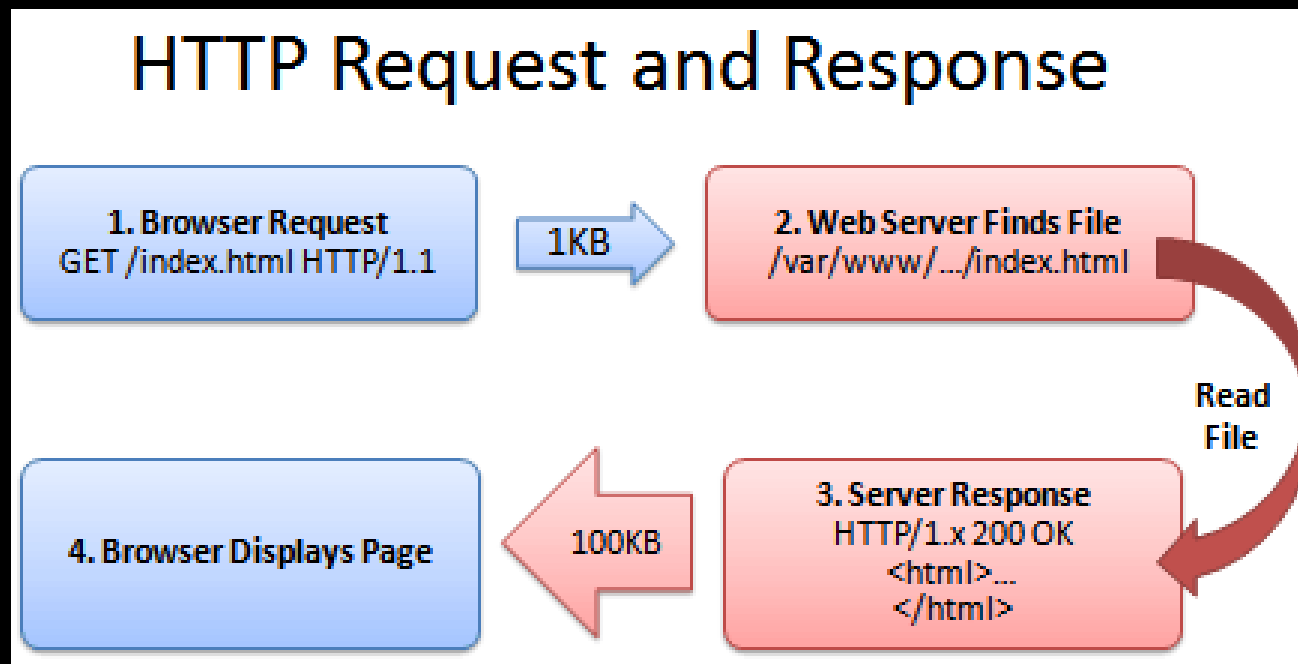
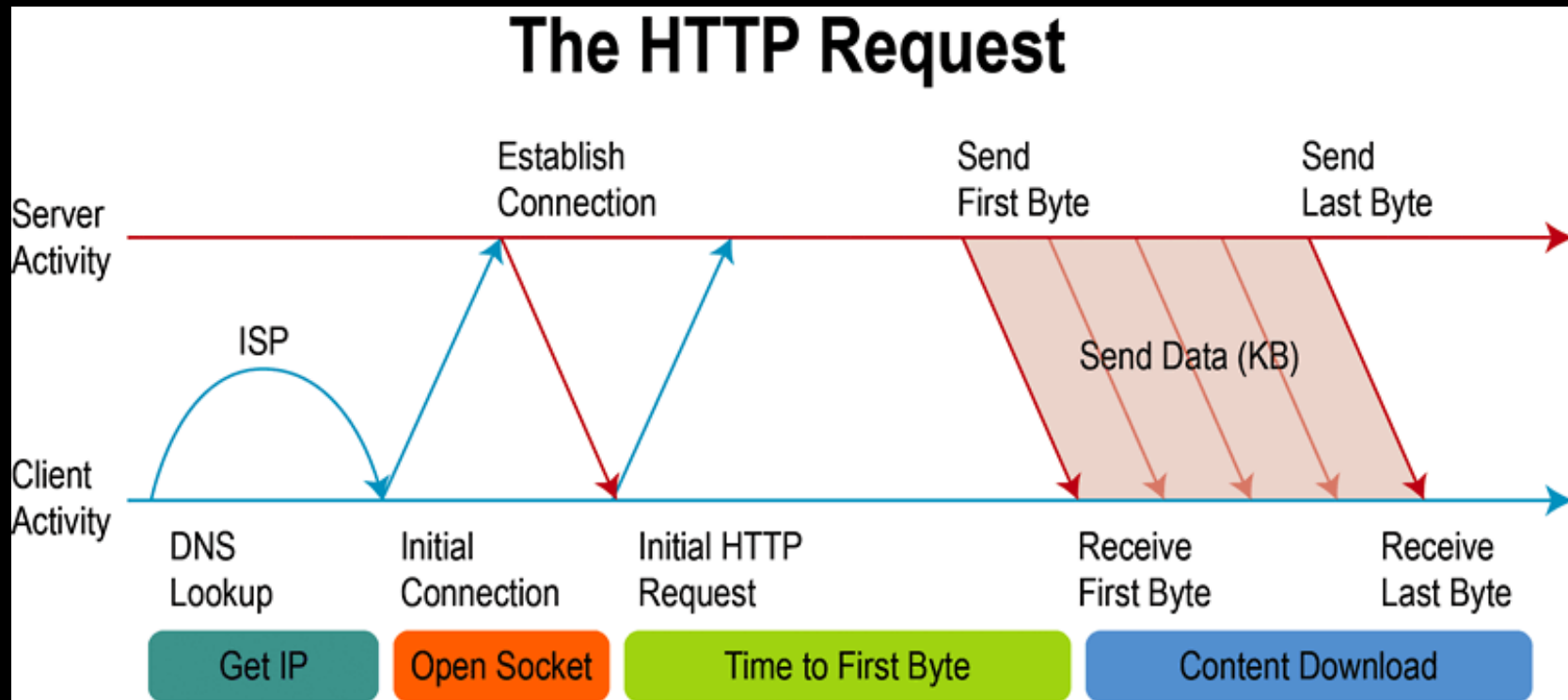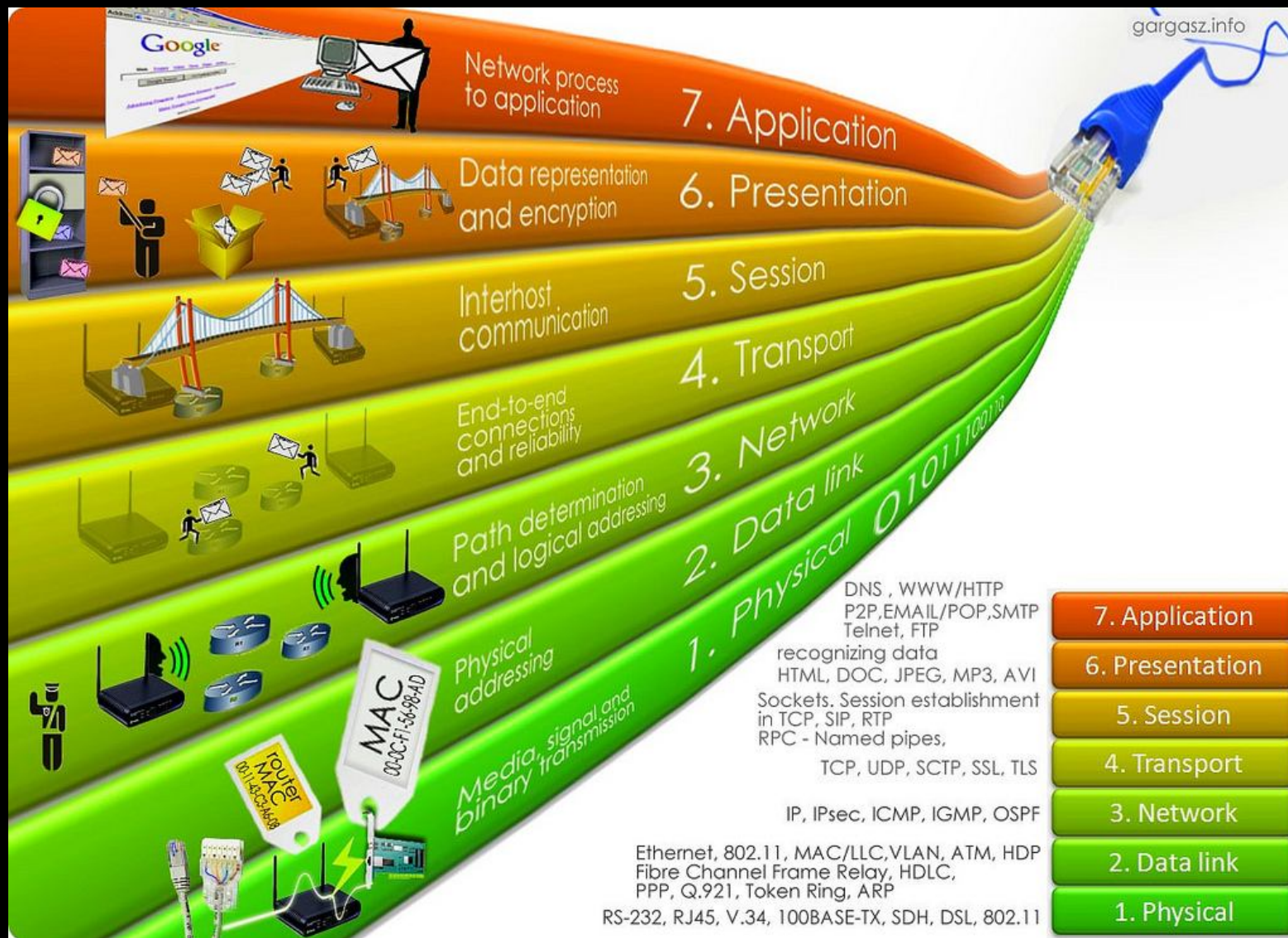Marios Karagiannopoulos / mariosk@gmail.com

Agenda

- The HTTP Request

- 7 Layers of OSI (Open System Interconnection)

- DNS Request

- HTTP Request Anatomy

- HTTP Request Methods, Headers

- HTTP Response Anatomy

- HTTP Response Codes

- Cookies

- Pipelining

- HTTP Analyzer

The HTTP Request (1)

HTTP Request and Response
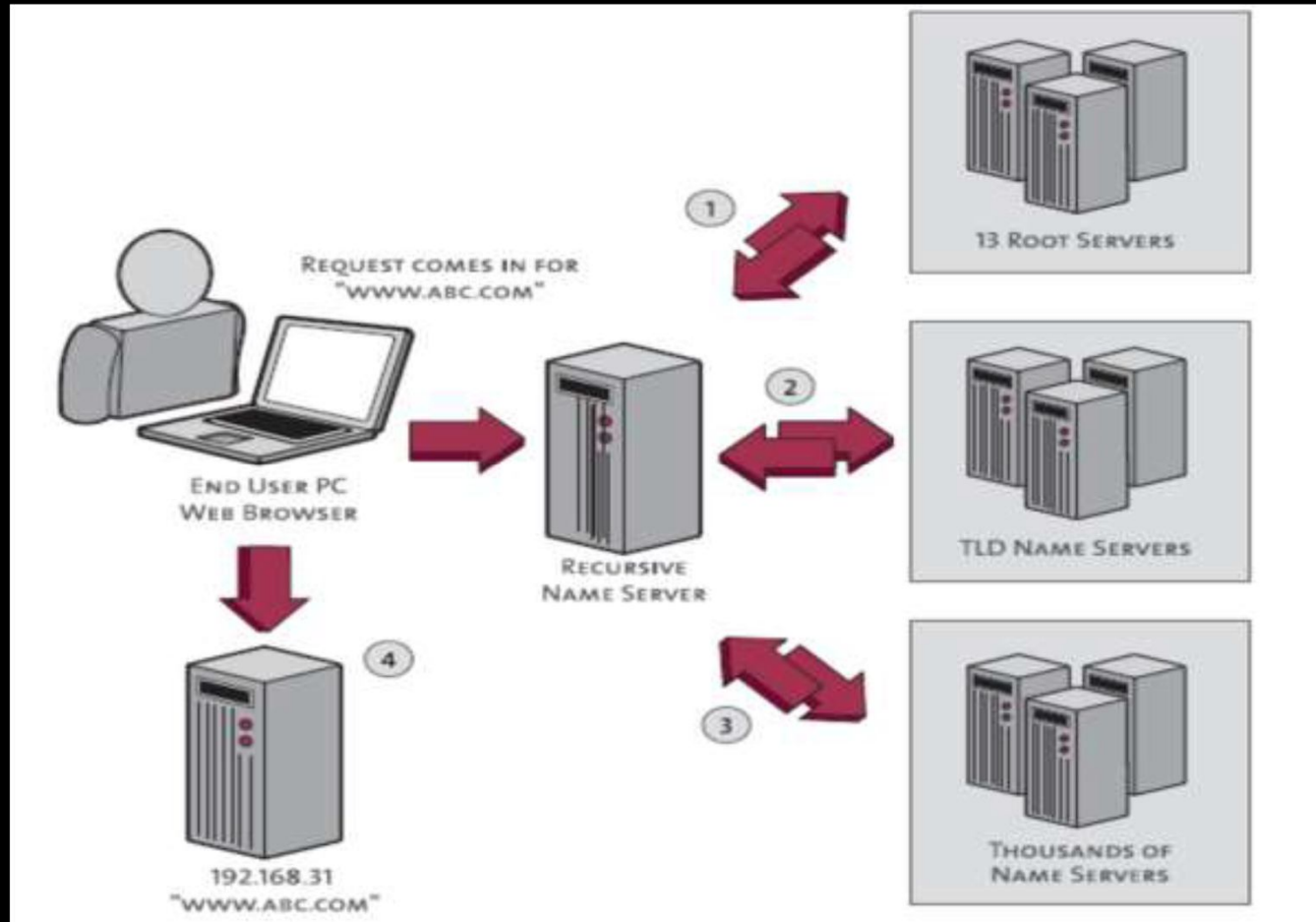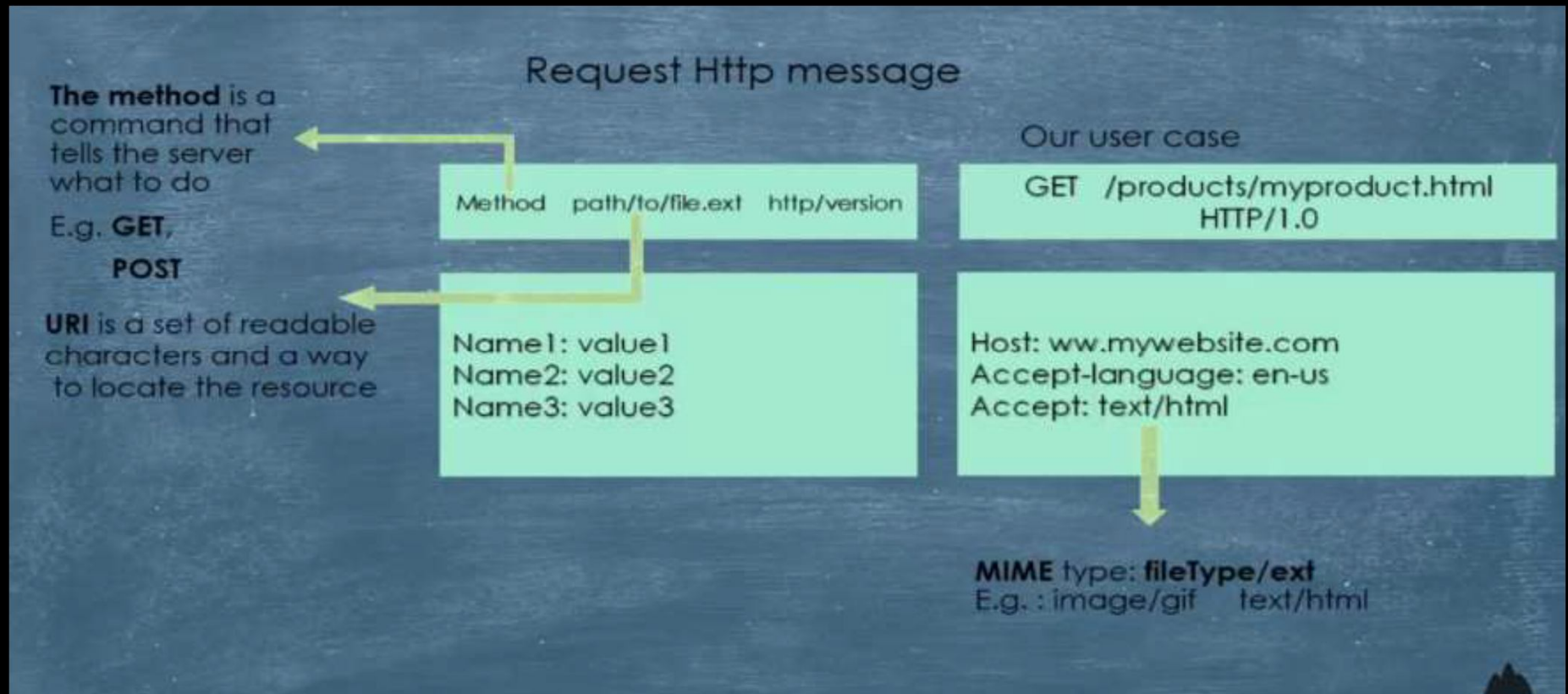
1. Browser Request
GET /index.html HTTP/1.1

1KB →

2. Web Server Finds File
/var/www/.../index.html

Read File

3. Server Response
HTTP/1.x 200 OK
<html>...
</html>

← 100KB

4. Browser Displays Page

The HTTP Request (2)

gargasz.info

Network process to application — 7. Application

Data representation and encryption — 6. Presentation

Interhost communication — 5. Session

End-to-end connections and reliability — 4. Transport

Path determination and logical addressing — 3. Network

2. Data link

Physical addressing — 1. Physical

Media, signal and binary transmission

MAC 00-0C-F1-56-98-AD

router MAC 00-11-43-C3-A6-08

DNS , WWW/HTTP
P2P,EMAIL/POP,SMTP
Telnet, FTP
recognizing data
HTML, DOC, JPEG, MP3, AVI
Sockets. Session establishment
in TCP, SIP, RTP
RPC - Named pipes,

TCP, UDP, SCTP, SSL, TLS

IP, IPsec, ICMP, IGMP, OSPF

Ethernet, 802.11, MAC/LLC,VLAN, ATM, HDP
Fibre Channel Frame Relay, HDLC,
PPP, Q.921, Token Ring, ARP
RS-232, RJ45, V.34, 100BASE-TX, SDH, DSL, 802.11

| 7. Application |
| 6. Presentation |
| 5. Session |
| 4. Transport |
| 3. Network |
| 2. Data link |
| 1. Physical |

DNS Request

HTTP Request Anatomy

HTTP Headers

Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Accept-Language	Request	The natural languages the client can handle
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Cookie	Request	Sends a previously set cookie back to the server
Date	Both	Date and time the message was sent
Upgrade	Both	The protocol the sender wants to switch to
Server	Response	Information about the server
Content-Encoding	Response	How the content is encoded (e.g., gzip)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Last-Modified	Response	Time and date the page was last changed
Location	Response	A command to the client to send its request elsewhere
Accept-Ranges	Response	The server will accept byte range requests
Set-Cookie	Response	The server wants the client to save a cookie

HTTP Request Methods

| Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

HTTP Response Anatomy
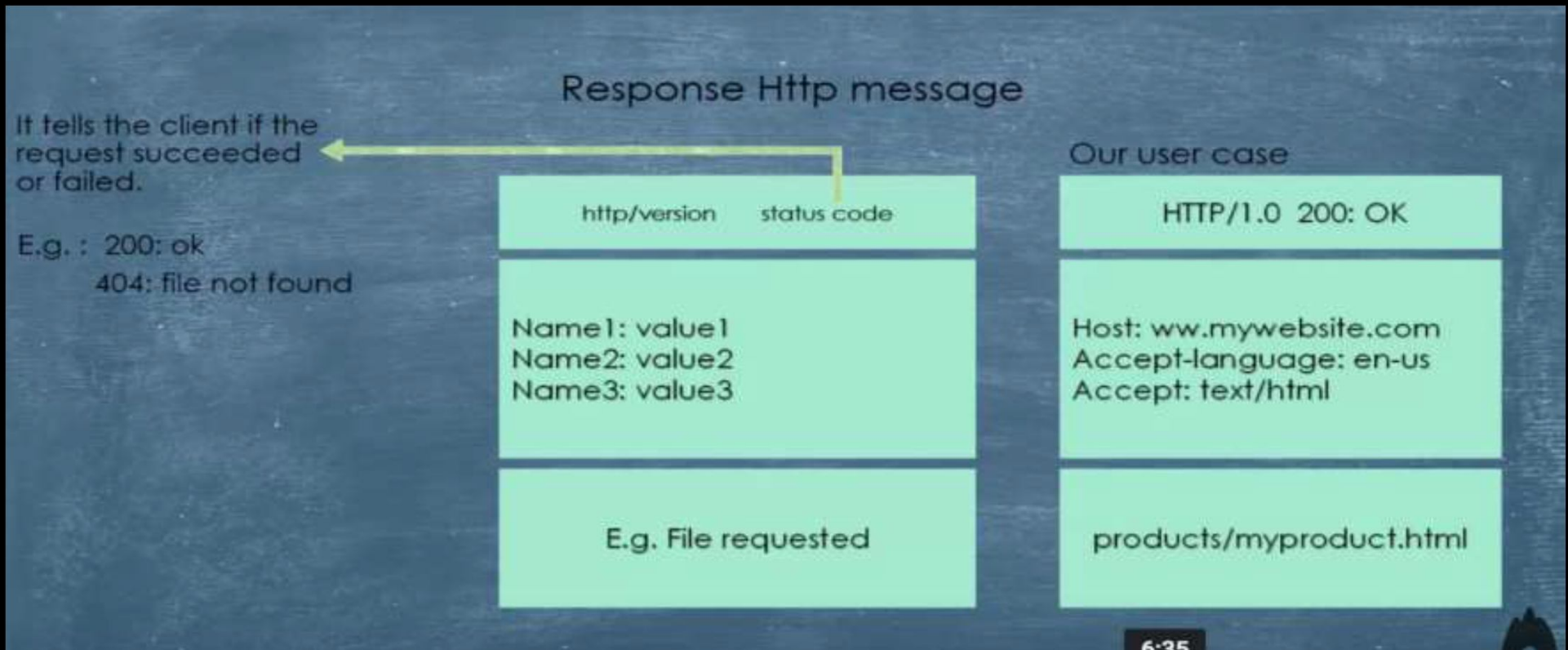
HTTP Response Codes

STATUS CODE	EXPLANATION
200 – OK	The request succeeded.
204 – No Content	The document contains no data.
301 – Moved Permanently	The resource has permanently moved to a different URI.
401 – Not Authorized	The request needs user authentication.
403 – Forbidden	The server has refused to fulfill the request.
404 – Not Found	The requested resource does not exist on the server.
408 – Request Timeout	The client failed to send a request in the time allowed by the server.
500 – Server Error	Due to a malfunctioning script, server configuration error or similar.

HTTP Cookies
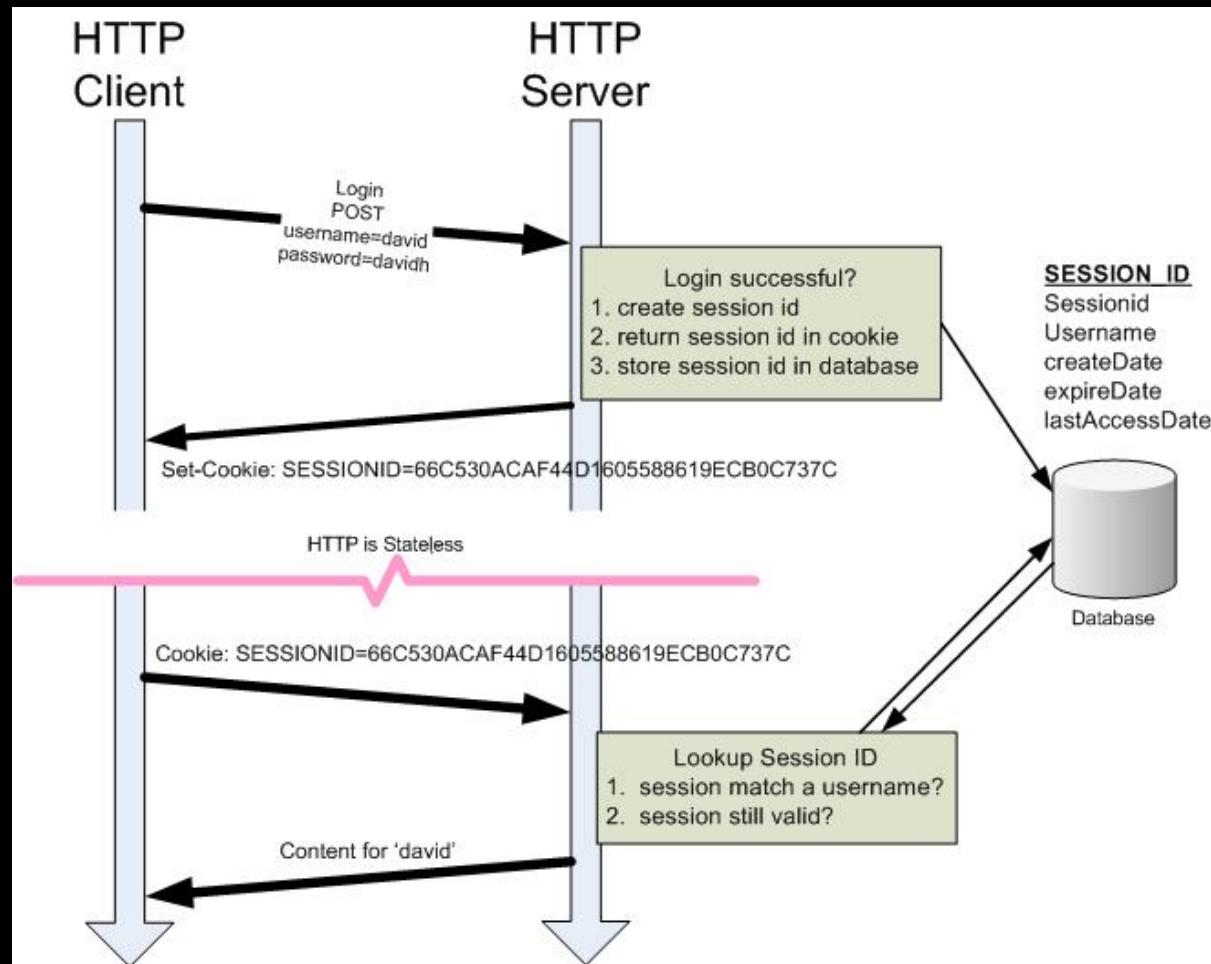
HTTP Pipelining (in HTTP/1.1)

(echo -en "GET /index.html HTTP/1.1\nHost: example.com\nConnection: keep-alive\n\nGET /index.html HTTP/1.1\nHost: example.com\n\n"; sleep 10) | telnet example.com 80
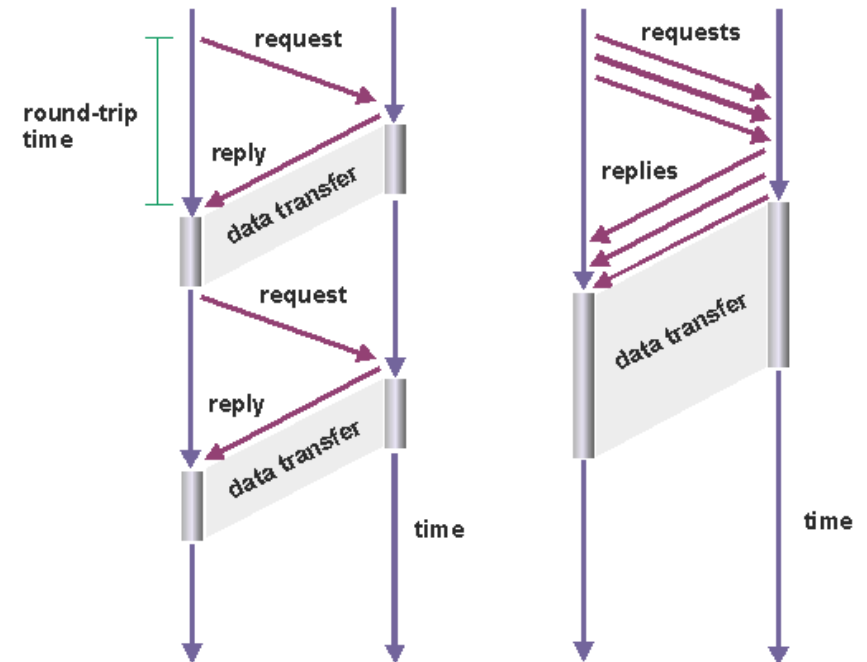
HTTP/2 is the future

HTTP/1.1 without pipelining: Each HTTP request over the TCP connection must be responded to before the next request can be made.

HTTP/1.1 with pipelining: Each HTTP request over the TCP connection may be made immediately without waiting for the previous request's response to return. The responses will come back in the same order.

HTTP/2 multiplexing: Each HTTP request over the TCP connection may be made immediately without waiting for the previous response to come back. The responses may come back in any order.

HTTP Analyzer