HTTP Deep Dive

André Luyer

#sf25eu

$ whoami

SharkFest'25 EUROPE

HTTP Deep Dive

André Luyer

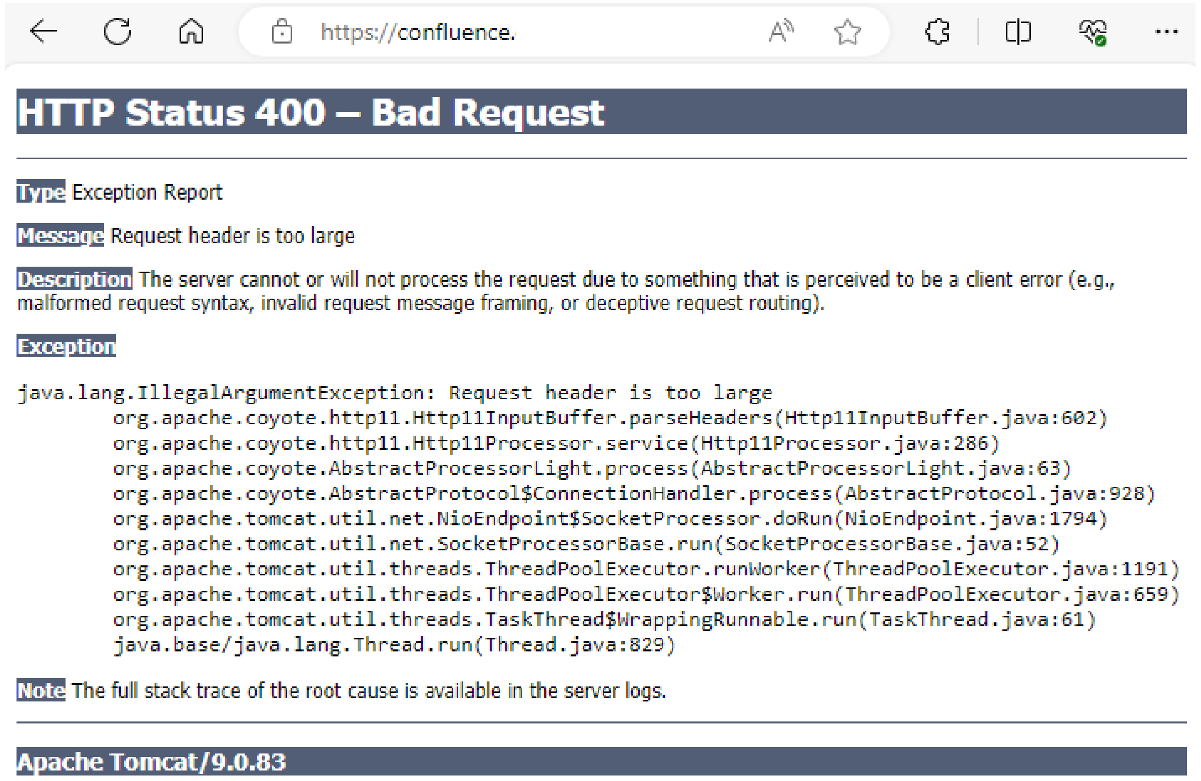
@AndreLuyer

Wireshark Users NL Meetup

- Performance, reliability, availability
- My focus is on application performance, so mainly the top level protocols
- Nowadays 99% of the cases it is HTTP
- Especially Cloud based apps

Agenda

- What is HyperText Transfer Protocol (HTTP)
- History
- HTTP/1
  - Header fields (headers)
  - Cookies
  - Caching
  - Response Codes
- HTTP/2 (TCP)
- HTTP/3 (QUIC)



HTTP Status 400 – Bad Request

Type Exception Report

Message Request header is too large

Description The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).

Exception

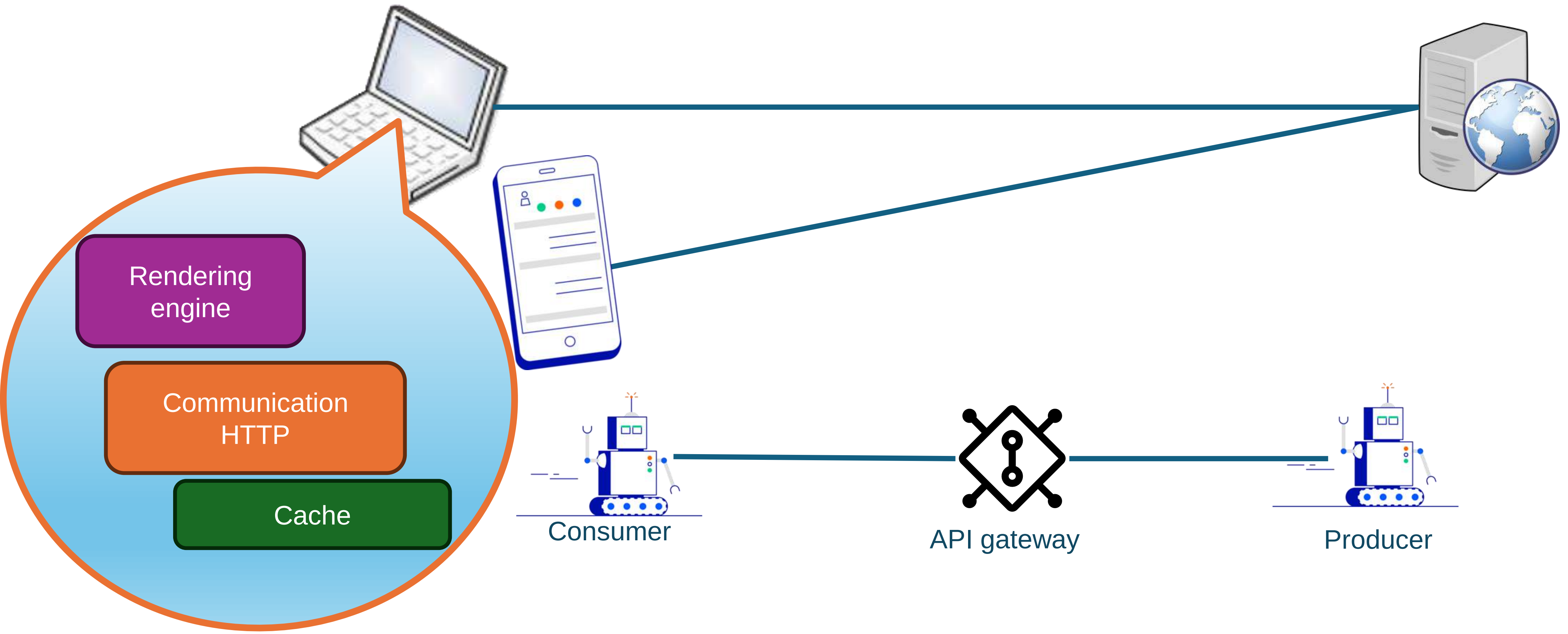
```
java.lang.IllegalArgumentException: Request header is too large
	org.apache.coyote.http11.Http11InputBuffer.parseHeaders(Http11InputBuffer.java:602)
	org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:286)
	org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:63)
	org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:928)
	org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1794)
	org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:52)
	org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1191)
	org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:659)
	org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
	java.base/java.lang.Thread.run(Thread.java:829)
```

Note The full stack trace of the root cause is available in the server logs.

Apache Tomcat/9.0.83

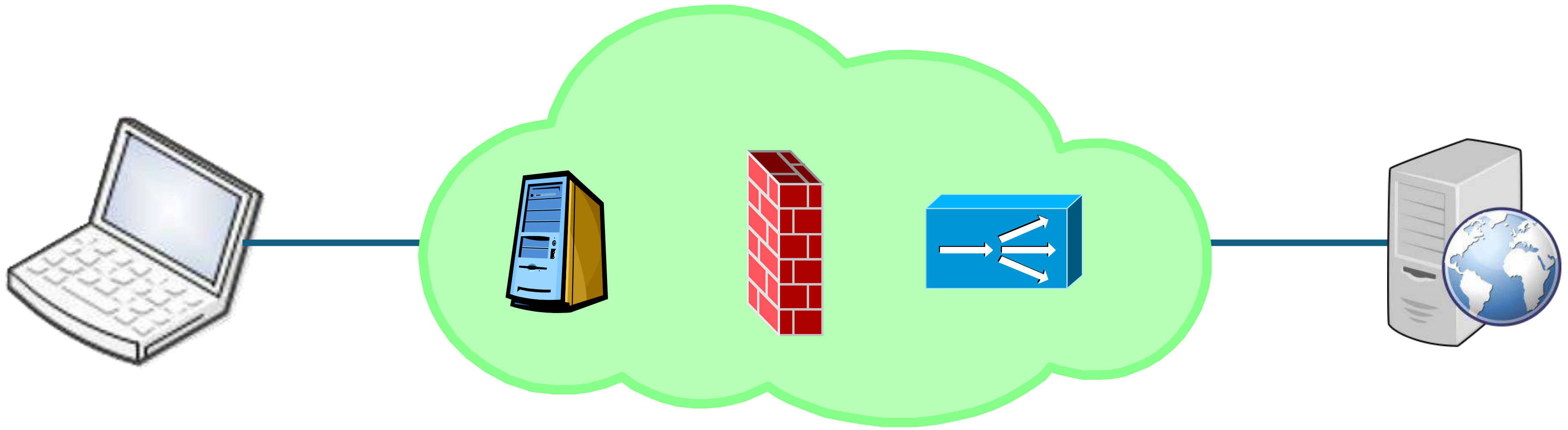
What is HyperText Transfer Protocol (HTTP)

A **T**ransfer **P**rotocol to exchange (download) **H**yper**T**ext, or any type of document, image, video, etc.



Network path

Typically, many devices in the network path

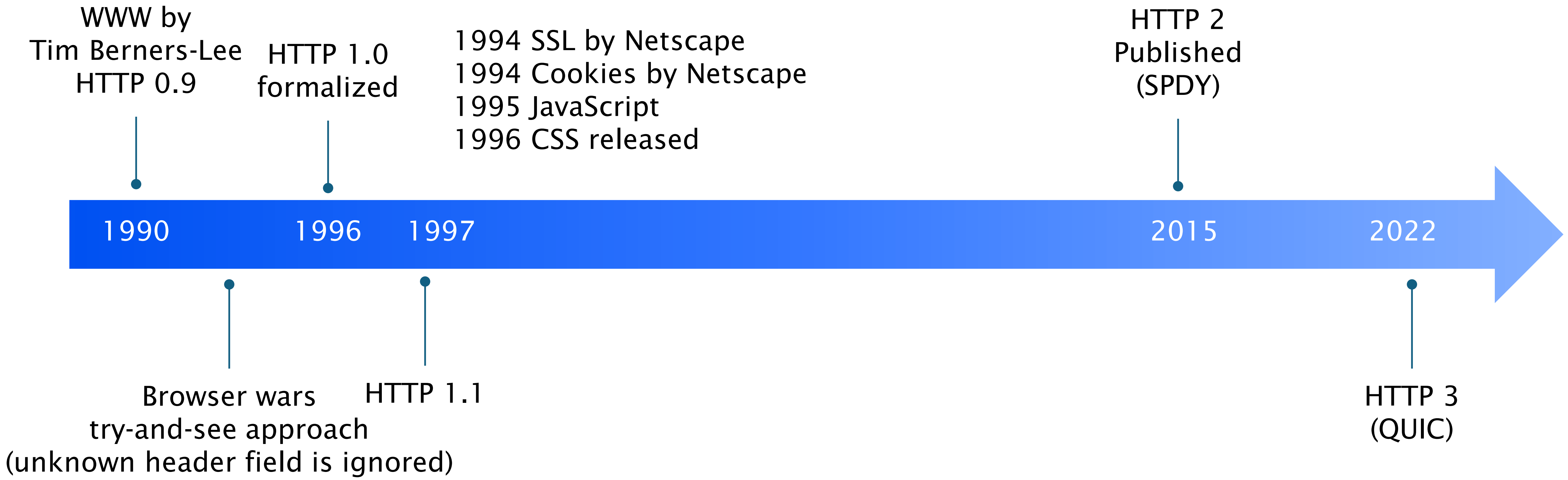


Many devices in the network path between browser and webserver
So called “middleboxes”
Such as: edge server, firewall, load balancer, (reverse) proxy, cache

HTTP/1.1

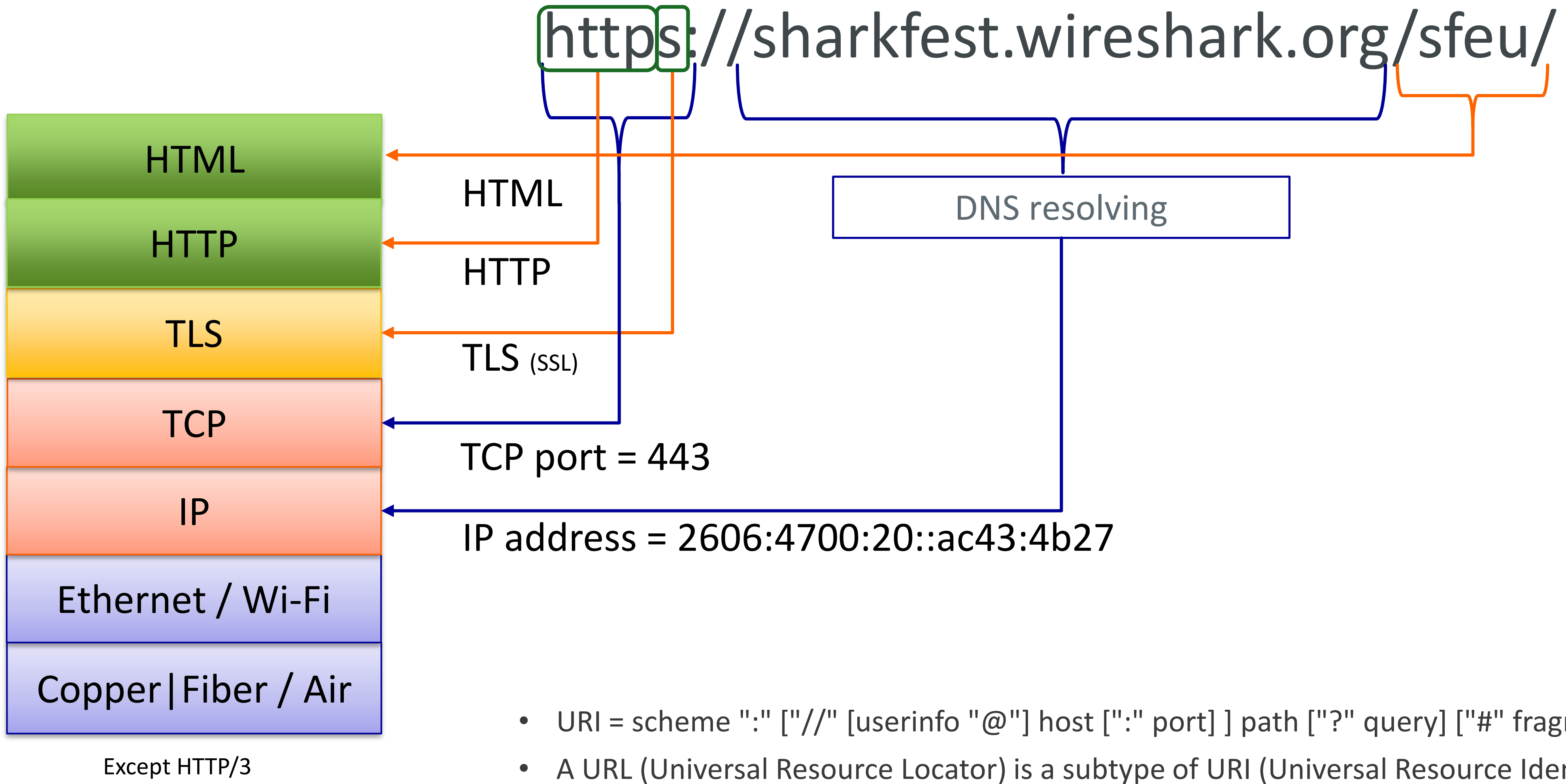
RFC9110

History



WWW = HTML + HTTP

URL to protocol stack



- URI = scheme ":" ["//" [userinfo "@"] host [":" port]] path ["?" query] ["#" fragment]
- A URL (Universal Resource Locator) is a subtype of URI (Universal Resource Identifier) containing both a resource and a protocol
- all URLs are URIs, but not all URIs are URLs

Terminology

HyperText Transfer Protocol (HTTP)

Client

(Consumer)

Initiates the connection(s)

Sends the request(s)

"User agent"

Browsers

Chrome, Firefox, Safari, Edge, …

Apps using a *browser engine* (like WebKit)

Postman, SoapUI

Uses a cache

Command line tools

Curl, wget, …

No caching

Company code

Crawler, IoT

Server

(Producer)

Accepts incoming connections

Sends response(s)

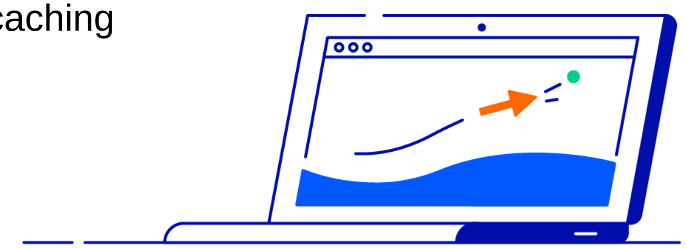
"Origin server"

Web servers

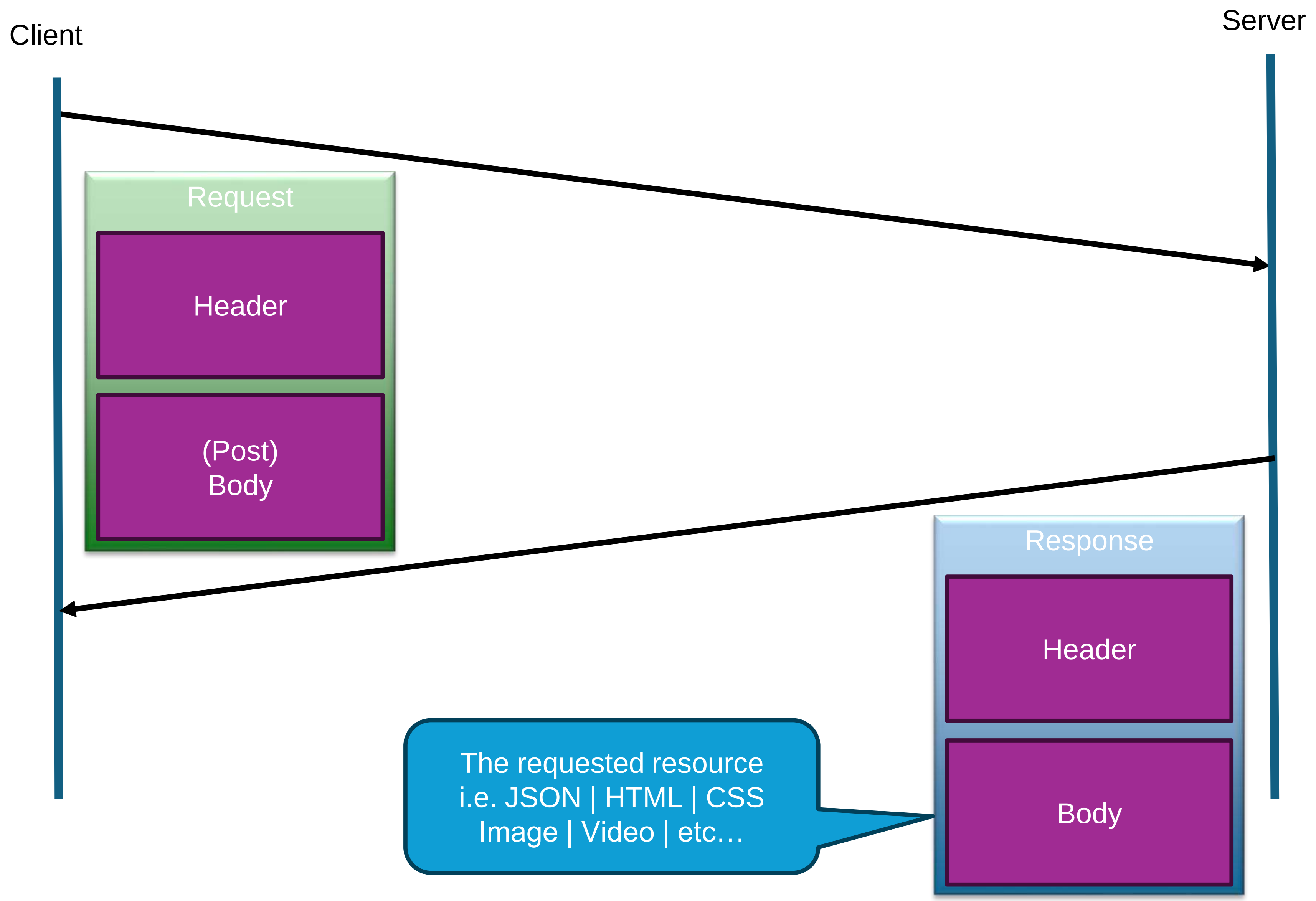
Nginx, Apache HTTP server, IIS, httpd, …

Company code, based on

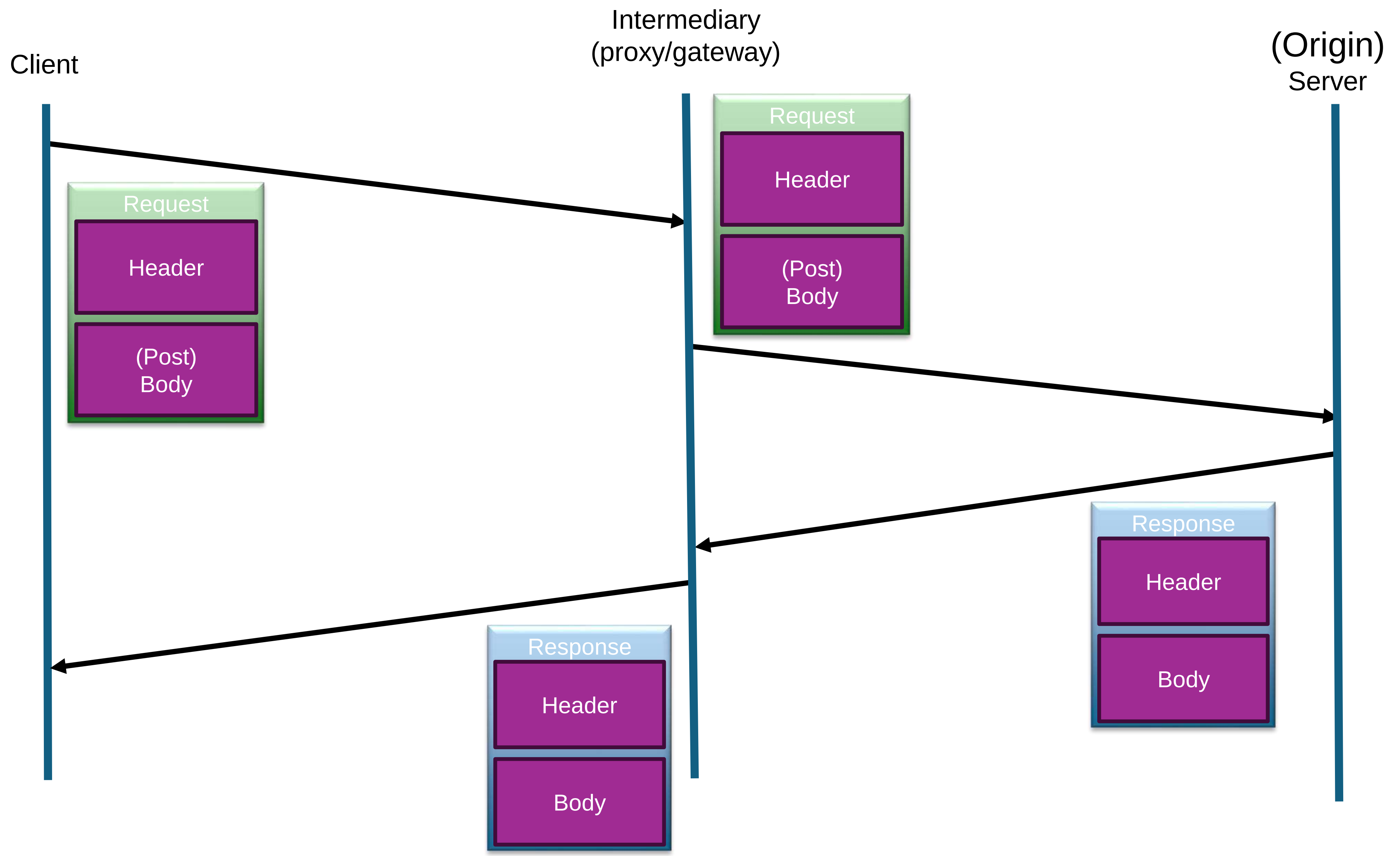
Spring Boot, Apache Tomcat, Jetty, …



Terminology (2)


Client
Server
Request
Header
(Post) Body
Response
Header
Body
The requested resource i.e. JSON | HTML | CSS Image | Video | etc…

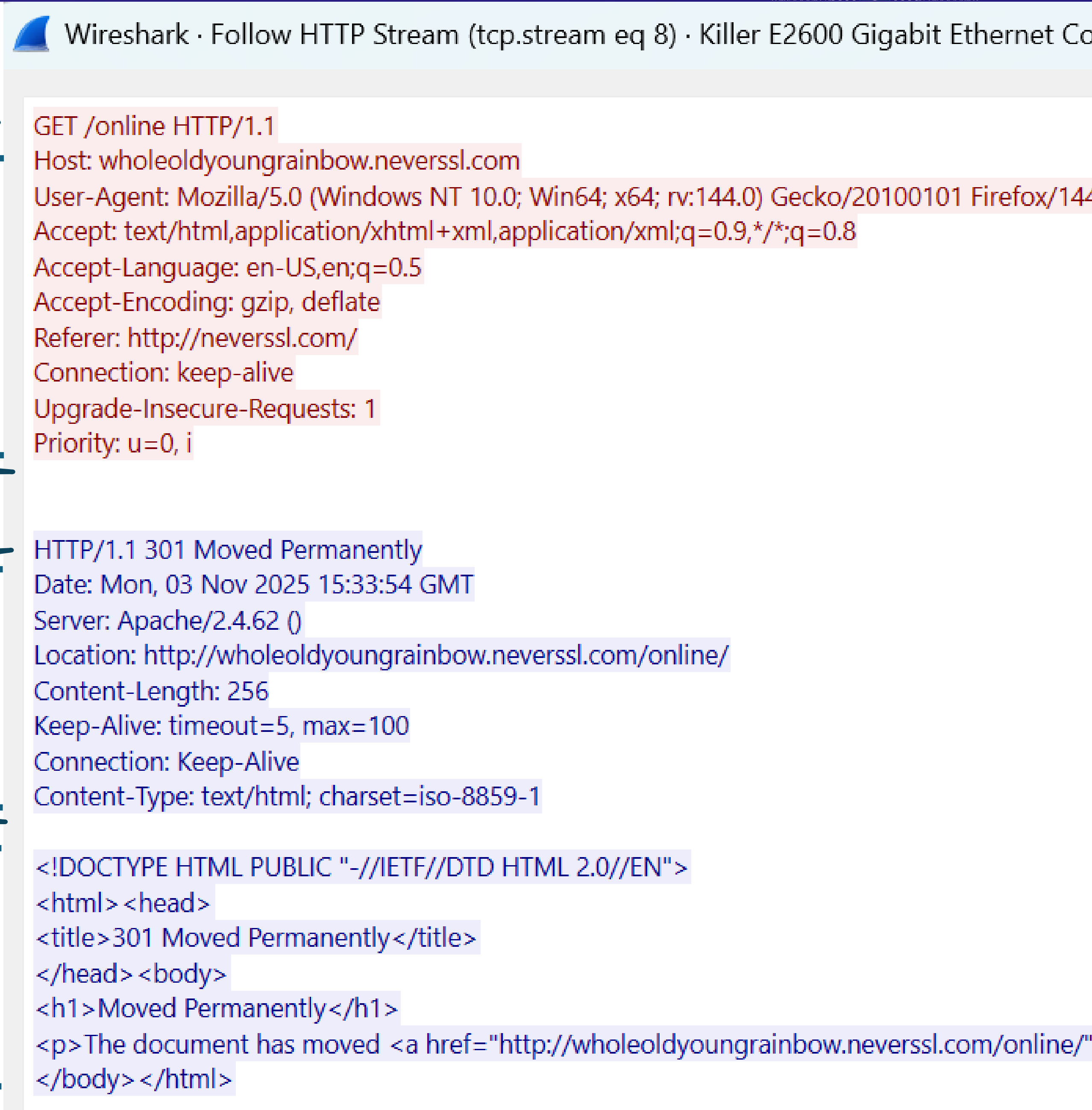
Terminology (3)



Header fields

a.k.a. headers

Header fields / Headers



Wireshark · Follow HTTP Stream (tcp.stream eq 8) · Killer E2600 Gigabit Ethernet Co

```
GET /online HTTP/1.1
Host: wholeoldyoungrainbow.neverssl.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:144.0) Gecko/20100101 Firefox/144
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://neverssl.com/
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=0, i


HTTP/1.1 301 Moved Permanently
Date: Mon, 03 Nov 2025 15:33:54 GMT
Server: Apache/2.4.62 ()
Location: http://wholeoldyoungrainbow.neverssl.com/online/
Content-Length: 256
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://wholeoldyoungrainbow.neverssl.com/online/"
</body></html>
```

Request-line

Header fields
name + ‘:’ + optional whitespace + value + CRLF

Blank line = end of request

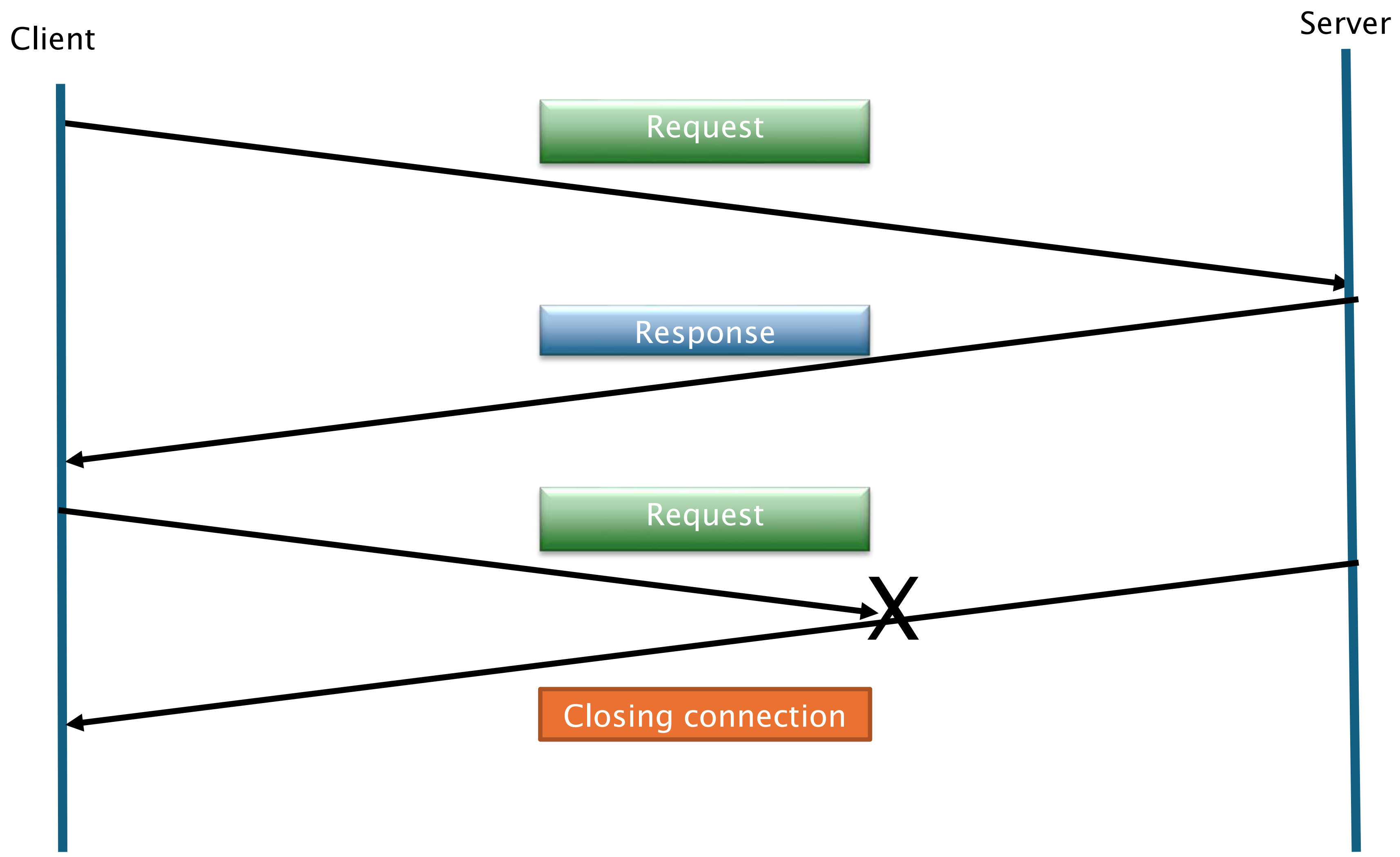
Status line

Header fields

Blank line = end of response

Response body

Connection: Close



Header fields

- Header field names are case-insensitive
- Unknown headers are ignored

  Add new feature… (try-and-see approach)

  Used to track path through intermediaries
- No limits of time or size in RFC (attack vector)
- Combine into 1 unique field using comma separated lists

  E.g. Accept, Cookie

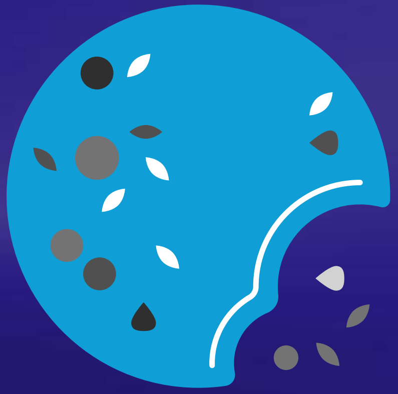
- Trailer fields are possible too

  (after “Trailer” header + chunked transfer encoding)

https://developer.mozilla.org/docs/Web/HTTP

Cookies

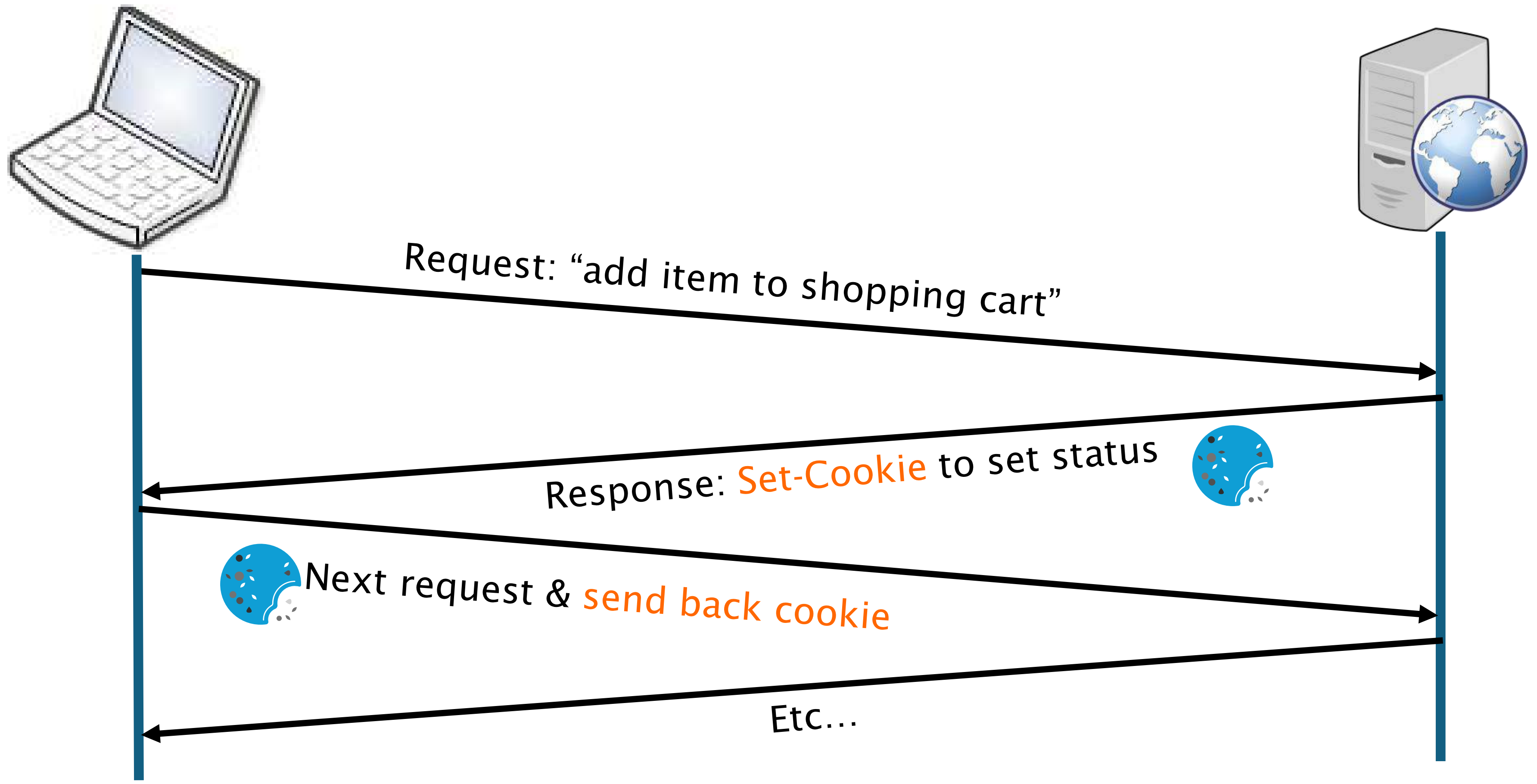
Since 1994



Cookies

A HTTP cookie is a (small) text in US-ASCII, send as a header field

For stateful information


Request: “add item to shopping cart”
Response: Set-Cookie to set status
Next request & send back cookie
Etc…

Cookie categories

HTTP itself is stateless, cookies adds “memory”

Categories:

- User preferences
- Session management (i.e. login status, shopping cart)
- User tracking ← General Data Protection Regulation (GDPR) EU regulation
  Opt-**in** required!!

Two types:

• Permanent cookies	deleted when on (or after) the specified date/time
• Session cookies	deleted when current session ends

Session cookie question

Session cookies   deleted when current session ends

Question: when does a session end?

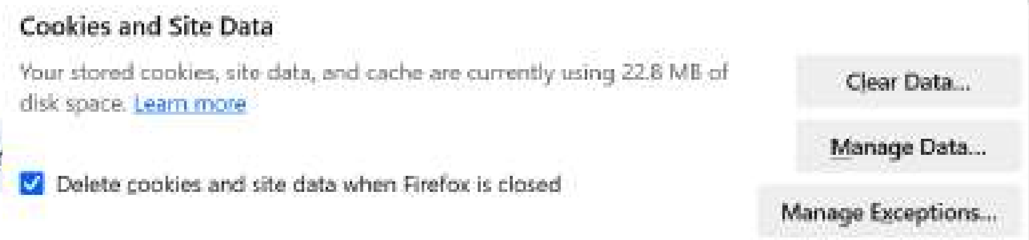
1. When the browser tab is closed
2. When the browser is closed
3. When I log out or reboot my laptop
4. Never

Cookie session

The browser defines what a ‘session’ is.

The session cookie is deleted when:

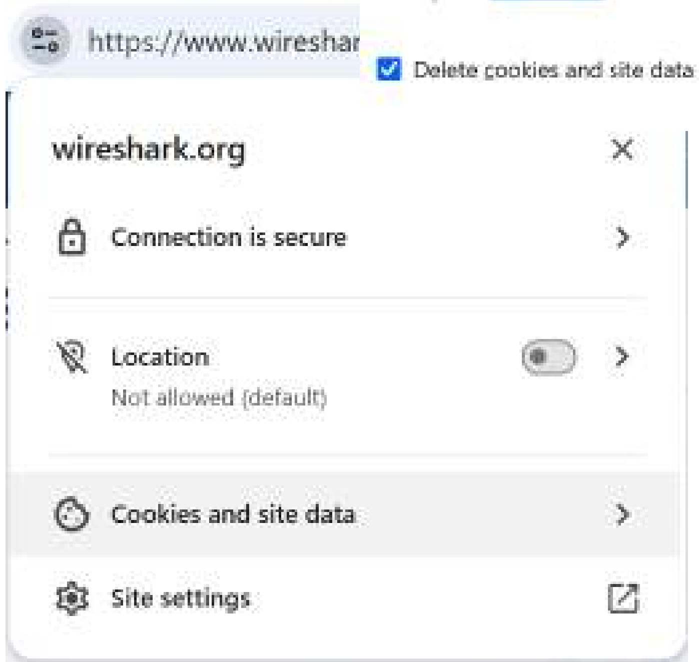
- Browser does not restore tabs when reopened → when browser is closed
- Browser does restore tabs when reopened → **never!**
- If website has a logout option → session cookies are usually deleted via logout-page



Cookies and Site Data

Your stored cookies, site data, and cache are currently using 22.8 MB of disk space. Learn more

[✓] Delete cookies and site data when Firefox is closed



https://www.wireshar

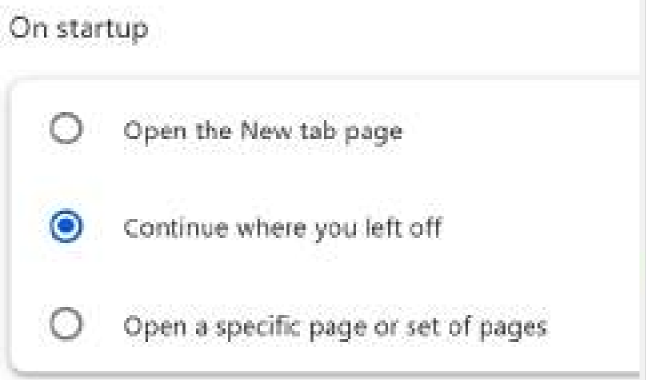
wireshark.org

Connection is secure

Location
Not allowed (default)

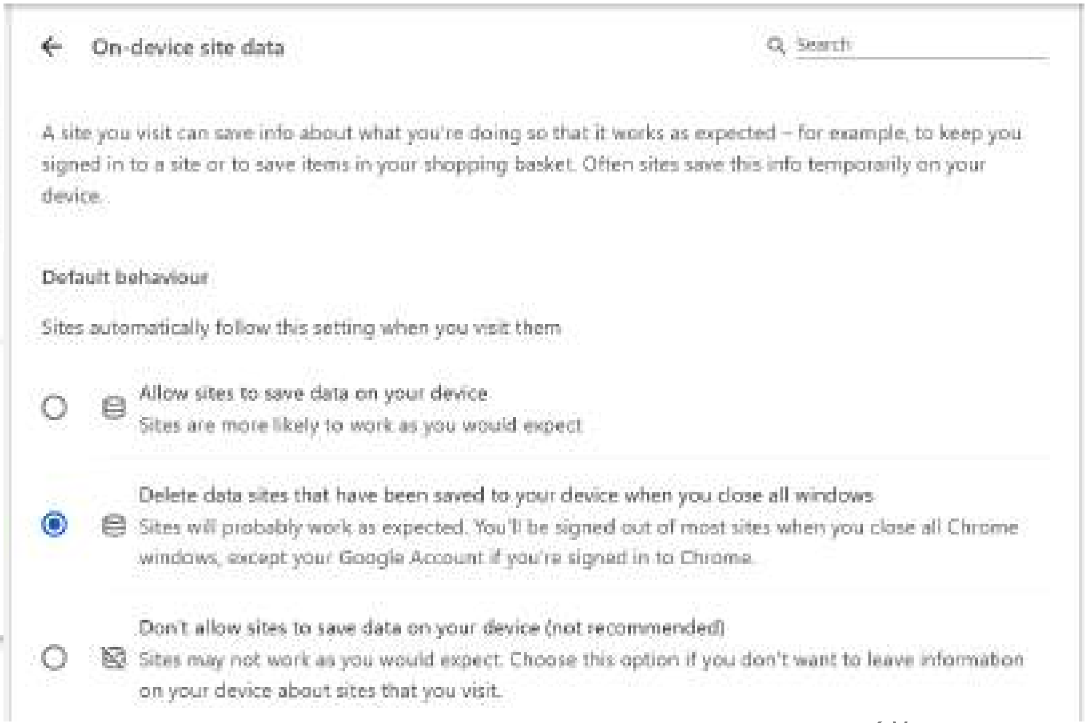
Cookies and site data

Site settings



On startup

- ( ) Open the New tab page
- [✓] Continue where you left off
- ( ) Open a specific page or set of pages



On-device site data

A site you visit can save info about what you're doing so that it works as expected – for example, to keep you signed in to a site or to save items in your shopping basket. Often sites save this info temporarily on your device.

Default behaviour

Sites automatically follow this setting when you visit them

- ( ) Allow sites to save data on your device
  Sites are more likely to work as you would expect
- [✓] Delete data sites that have been saved to your device when you close all windows
  Sites will probably work as expected. You'll be signed out of most sites when you close all Chrome windows, except your Google Account if you're signed in to Chrome.
- ( ) Don't allow sites to save data on your device (not recommended)
  Sites may not work as you would expect. Choose this option if you don't want to leave information on your device about sites that you visit.

Cookie Attributes

```
Set-Cookie: myCookie=some text here; Domain=wireshark.org; Path=/
```

Domain:	Defines the host – plus subdomains! – to which the cookie will be sent Default: the FQDN in the document URL
Path:	The path to match. "/" means the (document) root directory, thus *all* requests to that host Default: the directory the document resides in. So, for https://sub.domain.tld/path/to/file the path is "/path/to"
Expires:	Absolute end time (ISO-8601 format) A time in the past (or *invalid* time) causes the cookie to be deleted
Max-Age:	Time duration until the cookie expires, in seconds Zero to delete. Has precedence over Expires
HttpOnly:	Forbids JavaScript from accessing the cookie
Secure:	Send only when https is used

Expires / Max-Age: Not set = session cookie

See https://developer.mozilla.org/docs/Web/HTTP/Reference/Headers/Set-Cookie

Alternative: Web Storage API

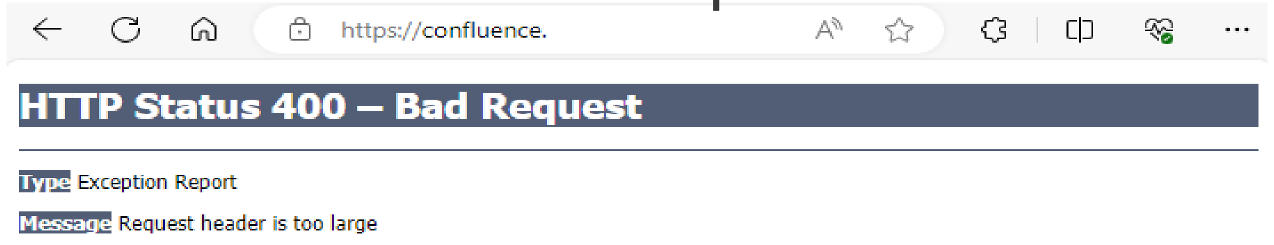
Mitigate 'Bad Request'

Many cookies due to many subdomains → oversized request

- As a developer
  - Sending Cookies
    - Set domain and path correctly; *with* subdomain and full path
  - Receiving Cookies
    - Accept larger request headers

User action needed =
at least one developer screwed up

- As a user
  1. Remove unneeded cookies via 'View site information'
  2. Configure browser *not* to restore sessions when reopening
  3. Use incognito mode or different profile



https://confluence.

HTTP Status 400 – Bad Request

Type Exception Report

Message Request header is too large

Caching

And compression

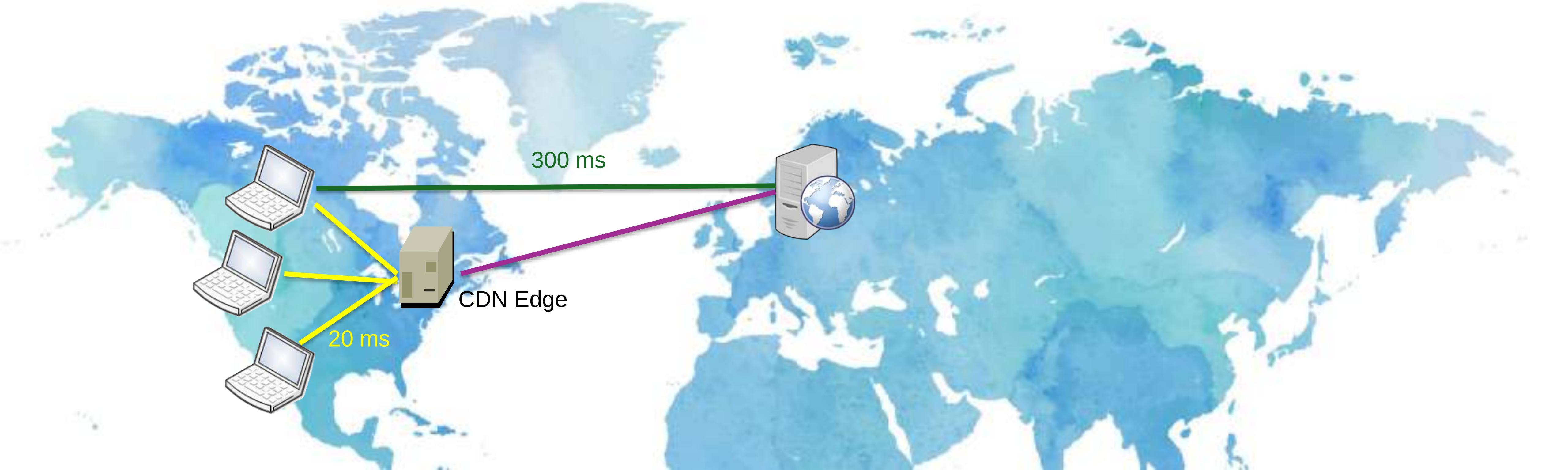
When use caching

Performance! End users don't want a sluggish app

The smaller the response (in bytes) the faster the download

- So a "not modified" response is better than the full document. Compression also
  - No network activity is even better; so when cached and fresh

Proxies, Edge (PoP) servers etc. can also cache contents



Cache

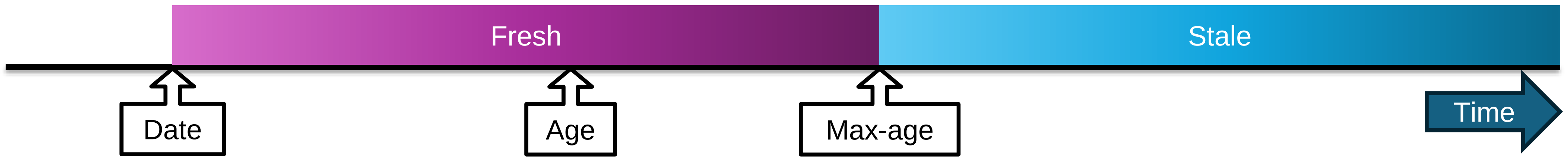
Types of caches

- Private caches
- Shared caches
  - Proxy caches
  - Managed caches (CDN, reverse proxy)

Most API consumers (clients) simply don’t implement any form of caching, but intermediary gateways may.

Goal of caching is performance gain! End users don’t want a sluggish app
So is compression

Caching



- First request: Date header contains time of generating the response, not arrival at client
- As long as the cache entry is **fresh**, **no** revalidation is done!
  - Except when user does a reload (F5 CMD-R) or a forced reload (Control-F5 Shift-CMD-R).
- When cache entry is **stale**, revalidation is attempted using request headers If-Modified-Since or If-None-Match
  - When HTTP 304 Not Modified is returned, the cached entry is returned and freshness updated (max-age)
  - When disconnected or not able to reach (origin) server:
    - Must-revalidate directive is not set: return stale entry!
    - Must-revalidate directive is set: return HTTP 504 Gateway Timeout
- When another client requests the same object, an intermediary cache returns the same entry with Age header
- The Vary header may be used to add dependency on specified request headers (i.e. multiple languages)

Cache-Control header

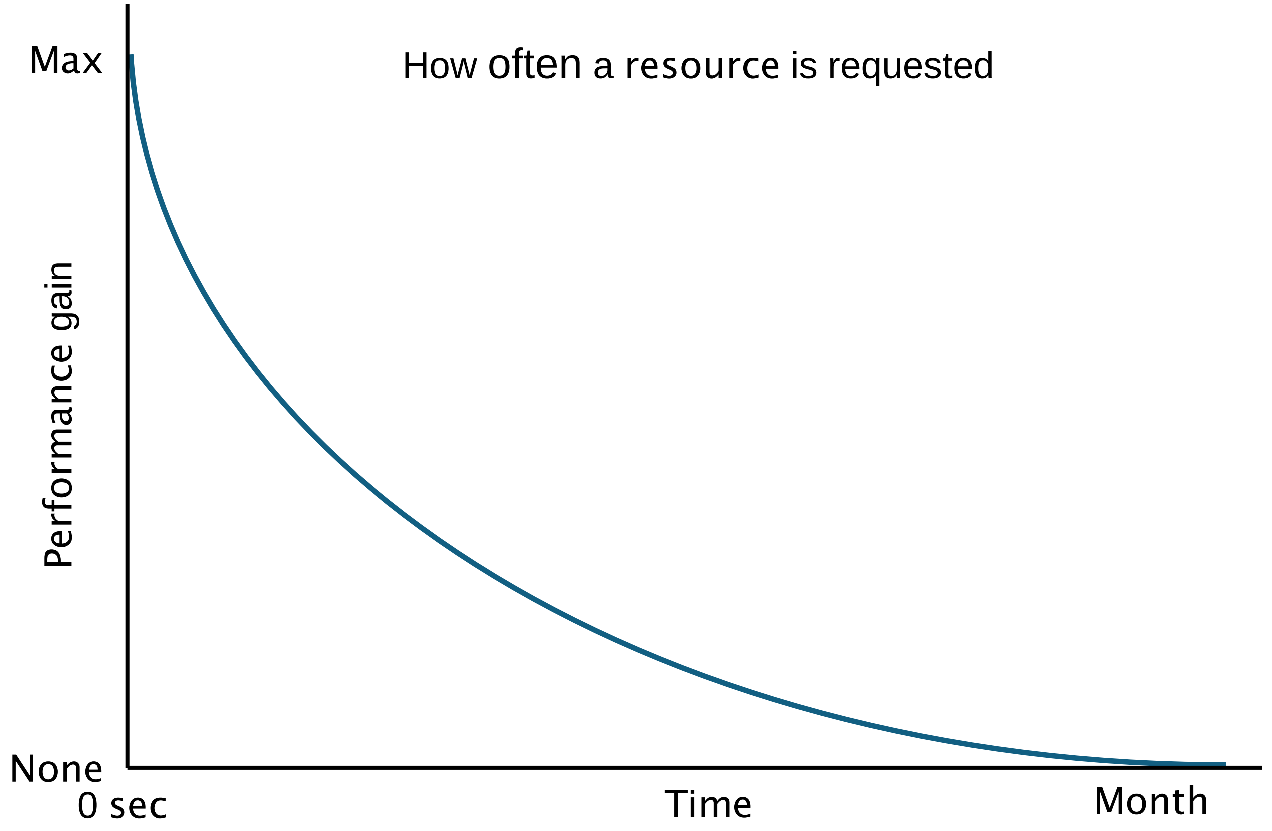
Everything can by specified by Cache-Control header, understood by ***all*** current UAs

Directives	Meaning
`max-age=N`	response remains fresh for N seconds
`s-maxage=N`	like max-age, but for shared cache
`no-cache`	mark stale immediately (so does cache), revalidate with origin server before reuse
`no-store`	don't store this response (and disable history)
`private`	can be stored only in a private cache (i.e. local cache in browsers)
`must-revalidate`	when stale, it must be validated with the origin server before reuse
`immutable`	to indicate the static resource is truly immutable (limited support yet)

To prevent caching use: `Cache-Control: no-store`
The directive `no-cache` is equal to `max-age=0, must-revalidate`
For more examples go to: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control

Caching – for how long?

- Using HTTP 304 (Not Modified) you can renew the freshness state – so only one request when stale
- Is it really necessary to keep object fresh in local (private) cache (storage) that long?
- ~~Removal of cached entry **not** possible via HTTP headers~~
- New since 2023 Clear-Site-Data header



Heuristic caching

Started with HTTP/1.0

When **no** or **invalid** caching headers are received, apply heuristic caching:

- Successful response is cacheable, except when data posted
- Fresh duration = 50% * (Date header (now) – Last-Modified header)

Exact logic depends on browser & version

Don’t rely on heuristic caching – always use Cache-Control header!

Response codes

Status codes

Or HTTP response codes

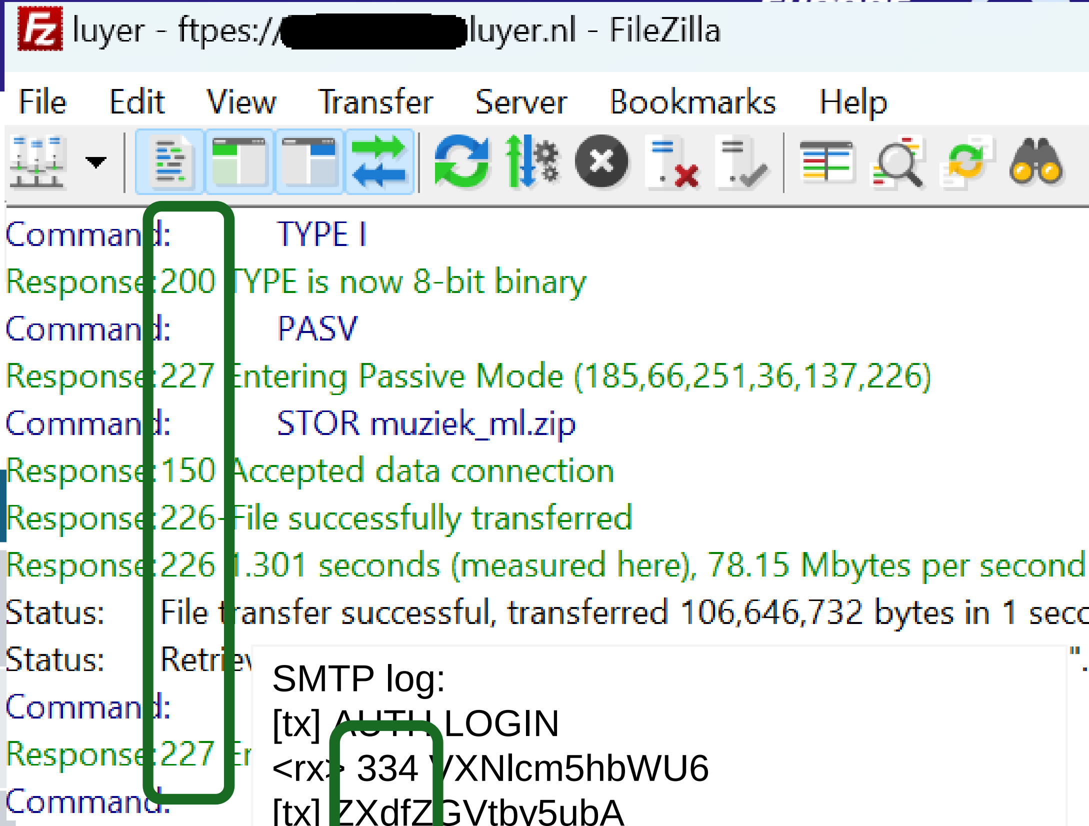
A status code is a three-digit integer code

The **first** digit of the status code defines the class of response

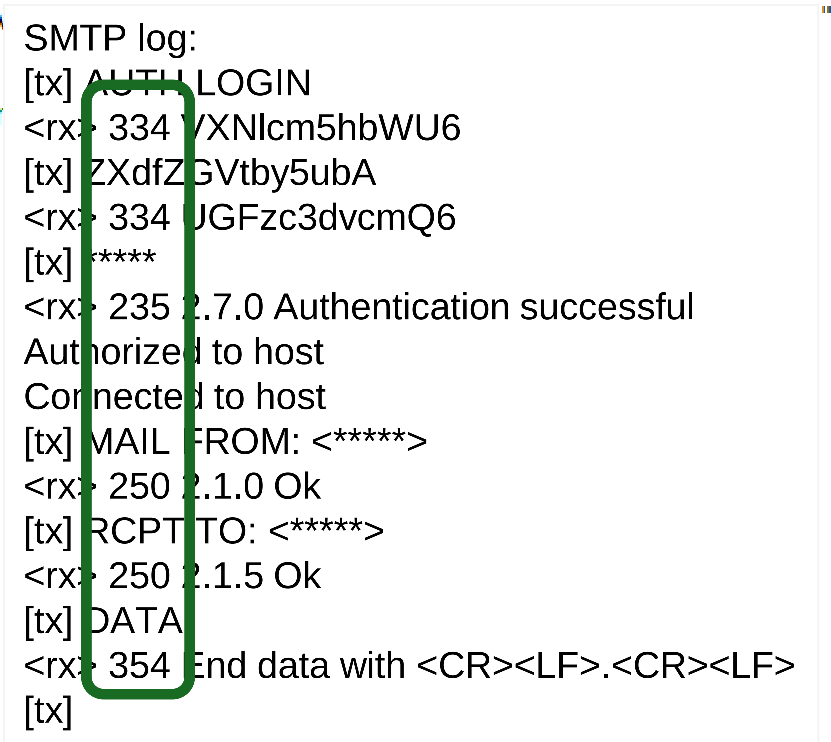
Code	Class	Meaning
1xx	Informational	the request was received, continuing process
2xx	Successful	the request was successfully received, understood, and accepted
3xx	Redirection	further action needs to be taken to complete the request
4xx	Client Error	the request contains bad syntax or cannot be fulfilled
5xx	Server Error	the server failed to fulfil request, but request itself was valid

https://datatracker.ietf.org/doc/html/rfc9110

https://developer.mozilla.org/docs/Web/HTTP/Status/



```
Command:    TYPE I
Response:   200 TYPE is now 8-bit binary
Command:    PASV
Response:   227 Entering Passive Mode (185,66,251,36,137,226)
Command:    STOR muziek_ml.zip
Response:   150 Accepted data connection
Response:   226-File successfully transferred
Response:   226 1.301 seconds (measured here), 78.15 Mbytes per second
Status:     File transfer successful, transferred 106,646,732 bytes in 1 seco
Status:     Retri
Command:
Response:   227 E
Command:
```



```
SMTP log:
[tx] AUTH LOGIN
<rx> 334 VXNlcm5hbWU6
[tx] ZXdfZGVtby5ubA
<rx> 334 UGFzc3dvcmQ6
[tx] *****
<rx> 235 2.7.0 Authentication successful
Authorized to host
Connected to host
[tx] MAIL FROM: <*****>
<rx> 250 2.1.0 Ok
[tx] RCPT TO: <*****>
<rx> 250 2.1.5 Ok
[tx] DATA
<rx> 354 End data with <CR><LF>.<CR><LF>
[tx]
```

Status codes – Informational class

Code	Short description	Meaning
100	Continue	typically on request; Expect header
101	Switching Protocols	response to an Upgrade request (HTTP/1.x only)
102	Processing	processing the request, but no response is available yet used by WebDAV
103	Early Hints	user agent may start preloading resources

Status codes – Successful class

Code	Short description	Meaning
200	OK	the request succeeded
201	Created	a new resource was created, typically after POST or PUT request
202	Accepted	the request has been received but not yet acted upon
203	Non-Authoritative Information	response may not be the same as from origin server
204	No Content	same as 200, but empty response body
205	Reset Content	client should reset the document view (i.e. clear form)
206	Partial Content	when client requested only part of a resource
207	Multi-Status	used by WebDAV
208	Already Reported	used by WebDAV
226	IM Used	a delta is returned instead of a full response

Status codes – Redirection class

Code	Short description	Meaning
300	Multiple Choices	the request has more than one possible response
301	Moved Permanently	the URL of this resource has been changed **permanently** SEO friendly; cache redirected URL
302	Found	the URL of this resource has been changed **temporarily** cache this URL, then follow redirect
303	See Other	Redirect plus change to GET method, i.e. prevent double POSTs
304	Not Modified	use the cached version and update caching info (like max-age)
307	Temporary Redirect	same as 302, but reuse (POST) method
308	Permanent Redirect	same as 301, but reuse (POST) method

Status codes – Client Error class (1)

Code	Short description	Meaning
400	Bad Request	server is unable or unwilling to process the request
401	Unauthorized	semantically this response means "unauthenticated"
402	Payment Required	experimental, treated as 400
403	Forbidden	refused, client is known but not authorized
404	Not Found	the server cannot find the requested resource
405	Method Not Allowed	request method is known, but not allowed on this URL
406	Not Acceptable	no match for request Accept header(s)
407	Proxy Authentication Required	similar to 401, but authentication needed by a proxy.
408	Request Timeout	waiting too long for a request (DDOS protection)
409	Conflict	request conflicts with the current state
410	Gone	like 404, but **permanent** and cacheable

Status codes – Client Error class (2)

Code	Short description	Meaning
411	Length Required	needed request Content-Length header field is absent
412	Precondition Failed	access to the target resource has been denied
413	Content Too Large	the request body (payload) is too large
414	URI Too Long	the length of the URL is too long
415	Unsupported Media Type	the payload format is in an unsupported format
416	Range Not Satisfiable	the requested range in Range header field cannot be fulfilled
417	Expectation Failed	the Expect request header field cannot be met by the server
421	Misdirected Request	the server is not able to produce a response for this redirect
422	Unprocessable Content	used by WebDAV
423	Locked	used by WebDAV
424	Failed Dependency	used by WebDAV

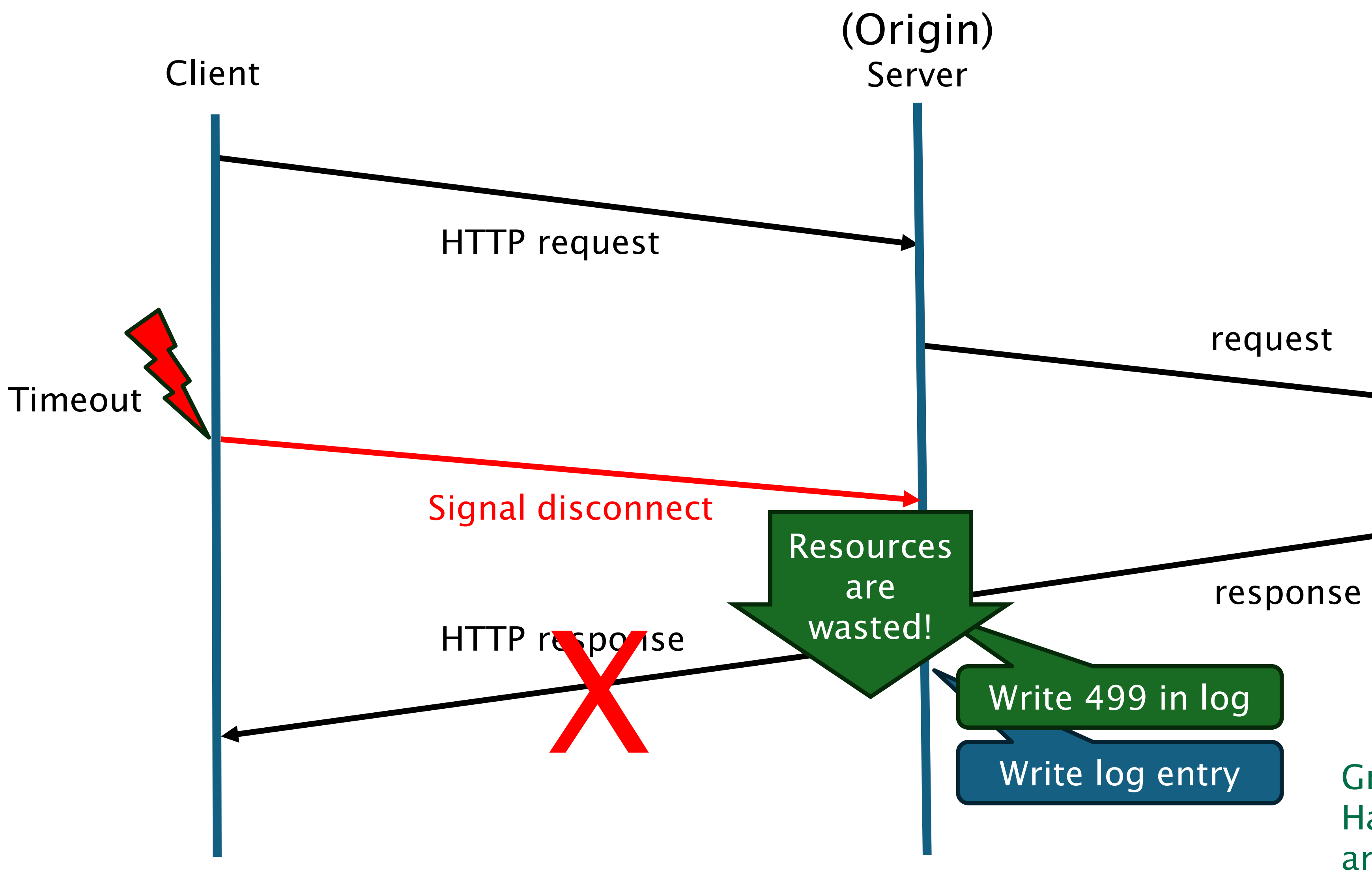
Status codes – Client Error class (3)

Code	Short description	Meaning
425	Too Early	protection against a potential replay attack
426	Upgrade Required	server refuses to perform the request using the current protocol
428	Precondition Required	this is intended to prevent the 'lost update' problem
429	Too Many Requests	rate limiting (DDOS protection)
431	Request Header Fields Too Large	either total request header or a header field too large
451	Unavailable For Legal Reasons	refused due to legal reasons

Pseudo status code:

Code	Short description	Meaning
499	Client Closed Request	client disconnected before response could be send

Timing and logging



Green coding:
Handle interrupt/signal/exception
and forward abort signal

Status codes – Server Error class

Thus, when the request itself was valid

Code	Short description	Meaning
500	Internal Server Error	a generic "catch-all" response
501	Not Implemented	request method is not recognized
502	Bad Gateway	received an invalid response from the upstream server
503	Service Unavailable	the server is not ready to handle the request
504	Gateway Timeout	did not get a response in time from the upstream server
505	HTTP Version Not Supported	the HTTP version used in the request is not supported
506	Variant Also Negotiates	the chosen variant is not a proper negotiation endpoint
507	Insufficient Storage	used by WebDAV
508	Loop Detected	used by WebDAV
510	Not Extended	
511	Network Authentication Required	generated by intercepting proxies that control network access

HTTP/2

RFC9113

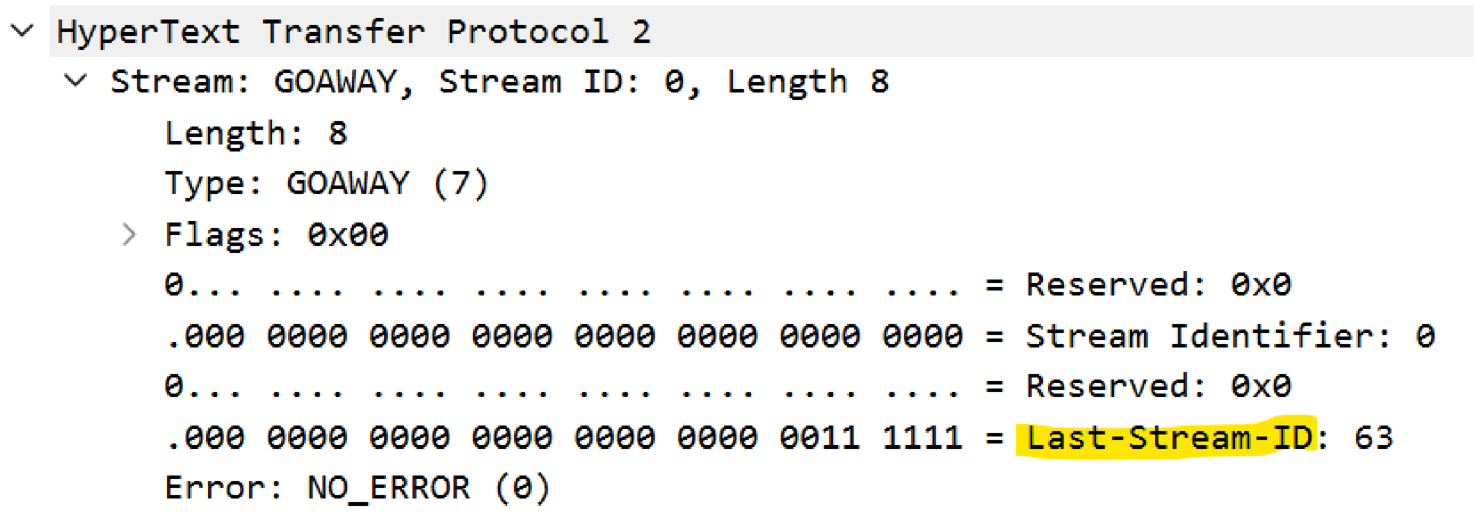
Features HTTP/2

- Use 1 TCP connection
  - no parallel TCP session needed (slow start, determine bandwidth)
- Use same port as HTTP/1
  - use ALPN in TLS handshake to determine version
  - Or ‘prior knowledge’ (h2c)
- Uses frames
- Server push
- “Connection: Close” header not allowed
- Header compression
  - Pseudo-Header Fields :method, :scheme, :path, :authority, :status
  - HPACK compression
    - Static & Dynamic Dictionary, Huffman Encoding
    - Many headers repeated in requests, use dynamic dictionary

GOAWAY frame

"Connection: Close" header not allowed, use GOAWAY instead

Last-Stream-ID: new streams in transit will be ignored
so no race-condition



```
∨ HyperText Transfer Protocol 2
  ∨ Stream: GOAWAY, Stream ID: 0, Length 8
      Length: 8
      Type: GOAWAY (7)
    > Flags: 0x00
      0... .... .... .... .... .... .... .... = Reserved: 0x0
      .000 0000 0000 0000 0000 0000 0000 0000 = Stream Identifier: 0
      0... .... .... .... .... .... .... .... = Reserved: 0x0
      .000 0000 0000 0000 0000 0000 0011 1111 = Last-Stream-ID: 63
      Error: NO_ERROR (0)
```

Features

Feature	HTTP/1.x	HTTP/2	HTTP/3
Cleartext (http)	Yes	Conditional	No
Encrypted (https)	Yes	Yes	Yes
Header compression	No	Yes (HPACK)	Yes (QPACK)
Server push	No	Yes	Yes

HTTP/3

RFC9114

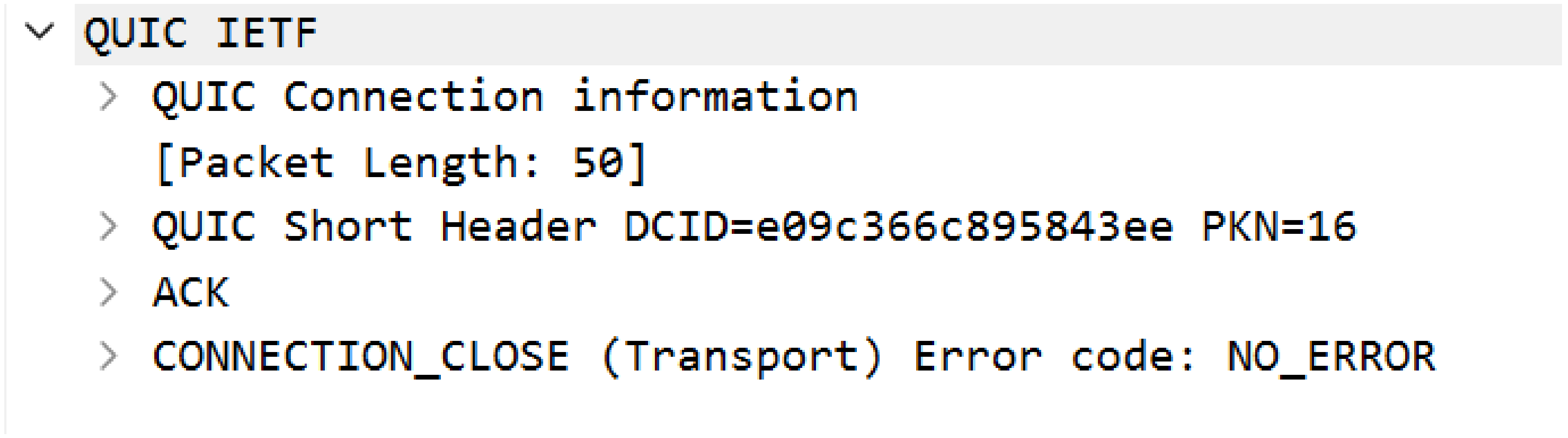
Features HTTP/3

Increased performance, specifically multiple objects

- Over UDP
- Session state by QUIC (not TCP)
  - Allows change in network, e.g. Wi-Fi – mobile, without loosing connection (IP address change)
- Faster, QUIC combining TCP/TLSv1.3
- No more TCP's head-of-line blocking
- Stream prioritization more flexible and efficient
- Connection 0-RTT resumption
- QPACK compression
- PUSH_PROMISE frame

Connection Close

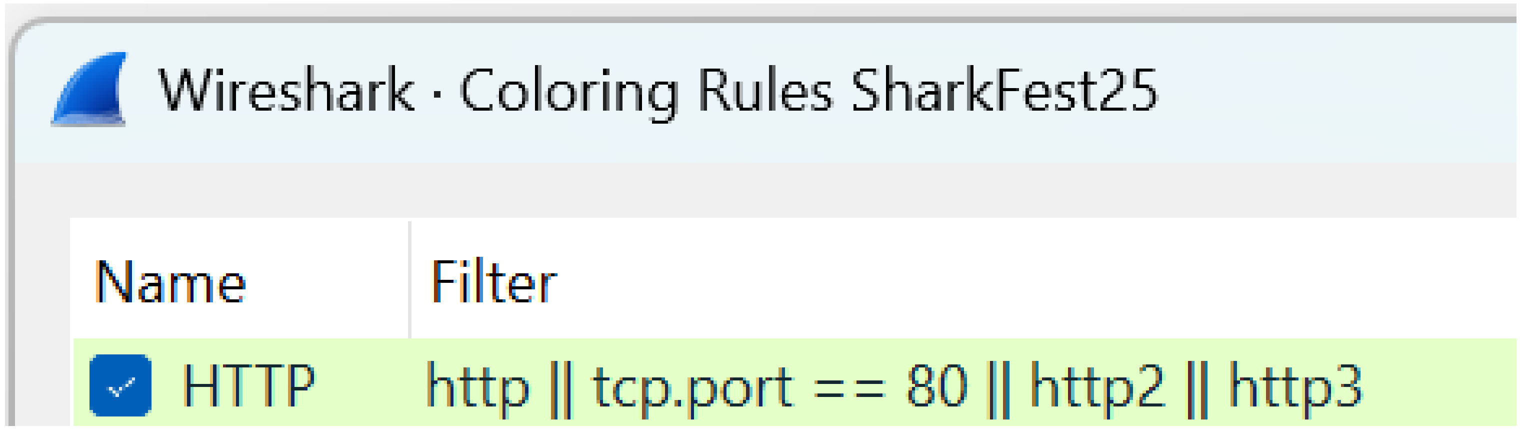
Close connection by stream



```
∨ QUIC IETF
  > QUIC Connection information
    [Packet Length: 50]
  > QUIC Short Header DCID=e09c366c895843ee PKN=16
  > ACK
  > CONNECTION_CLOSE (Transport) Error code: NO_ERROR
```

Colorfilter

Update your *existing* Coloring Rules



Wireshark · Coloring Rules SharkFest25

Name	Filter
✓ HTTP	http || tcp.port == 80 || http2 || http3

Add this

(My PR went into the master branch today, so the next release will have the updated default color rule)

Summary

Try-and-see approach still in effect. Applies to:

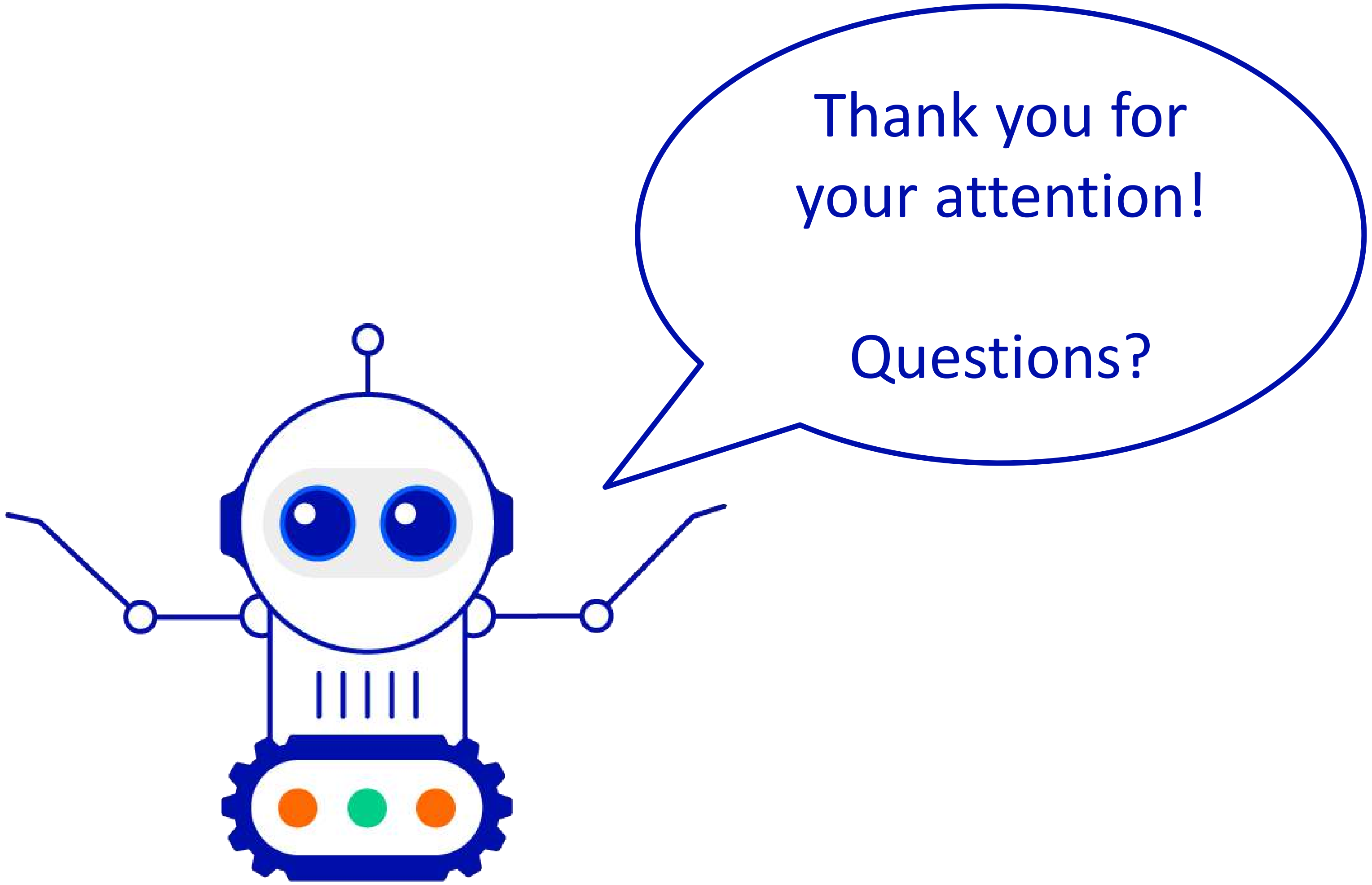
- Header fields a.k.a. headers
- Cookies
- Status codes

Ignored when not understood, status code treated as x00

You can use that to make your traffic easy to find in pcap!

HTTP 499 – abort processing when client disconnects

HTTP/2 & HTTP/3 performance improvements on front-end, binary headers


Thank you for your attention!

Questions?

Useful links – protocol descriptions

RFCs		
Overview:	https://developer.mozilla.org/docs/Web/HTTP/Resources_and_specifications	
HTTP:	https://datatracker.ietf.org/doc/html/rfc9110	
HTTP/2:	https://datatracker.ietf.org/doc/html/rfc9113	
HTTP/3:	https://datatracker.ietf.org/doc/html/rfc9114	
TLS:	https://datatracker.ietf.org/doc/html/rfc8446	
TCP:	https://datatracker.ietf.org/doc/html/rfc9293	(First RFC 793 September 1981)
IPv4:	https://datatracker.ietf.org/doc/html/rfc791	(First RFC 760 January 1980)
IPv6:	https://datatracker.ietf.org/doc/html/rfc8200	
DNS:	https://datatracker.ietf.org/doc/html/rfc1035	(First RFC 882 November 1983)

History: https://www.computerhistory.org/internethistory/
https://en.wikipedia.org/wiki/HTTP_cookie

Manual client

Unencrypted:

```
telnet hostname 80
curl –v telnet://hostname:80
```

<type or post the request header here>

Encrypted:

```
openssl s_client -crlf -quiet -brief -connect hostname:443
```

<type or post the request header here>

```
echo "GET / HTTP/1.1
Host: hostname
Connection: Close
" | openssl s_client -crlf -quiet -brief -connect hostname:443
```

Curl tips & tricks

In pipelines and scripts

Curl command: `curl [options] URL [options2] [URL2] …`

Useful options:

Short	Long	Description
-s	--silent	Mute curl, don't clog the log file with progression bar lines
-S	--show-error	… but do show errors
-m	--max-time	Limit the maximum duration
	--connect-timeout	Limit the time to set up the connection
	--compressed	Accept compressed content
	--resolve	Use specified address, instead of hostname in URL, to connect
-c	--cookie-jar	Save cookie(s) to specified file
-b	--cookie	Send cookie(s) specified on command line or in file

Example: `curl -sSm5 …`

https://curl.se/docs/manpage.html

HTTP methods in curl

Method	Command line
GET	`curl URL`
POST	`curl --data "post data" URL`
HEAD	`curl --head URL`
PUT	`curl --request PUT --data "post data" URL`
PATCH	`curl --request PATCH --data "post data" URL`
DELETE	`curl --request DELETE --data "post data" URL`
CONNECT	`curl --proxy proxy:port URL`
OPTIONS	`curl --request OPTIONS URL`
TRACE	`curl --request TRACE URL`

The `request` option replaces the GET/POST word in the request-line without altering Curl’s behaviour