

# HTTP

## MPRI 2.26.2: Web Data Management

---

Antoine Amarilli



# HTTP (HyperText Transfer Protocol), layer 7

- The **World Wide Web** (WWW)
- **Protocol** for Web browsing

→ Summary: we have

- the **client machine**
- a **client software**: the **Web browser**
- a **server machine**
- a **server software**: the **Web server**
- a **reliable, encrypted** communication channel

# Overview

- Standardized by the **Internet Engineering Task Force** (IETF) and the **World Wide Web Consortium** (W3C)
- Official standard: RFC 2616 (114 pages, 1999, + followups)

# Overview

- Standardized by the **Internet Engineering Task Force** (IETF) and the **World Wide Web Consortium** (W3C)
- Official standard: RFC 2616 (114 pages, 1999, + followups)
- **Extensions** : WebSockets, new headers, etc.

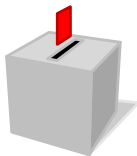
# Overview

- Standardized by the **Internet Engineering Task Force** (IETF) and the **World Wide Web Consortium** (W3C)
- Official standard: RFC 2616 (114 pages, 1999, + followups)
- **Extensions** : WebSockets, new headers, etc.
- New versions: **HTTP/2** and **HTTP/3**

## POLL: HTTP/2

Which proportion of websites uses HTTP/2 today?

- **A:** less than 25%
- **B:** 25%–50%
- **C:** 50%–75%
- **D:** over 75%



## POLL: HTTP/2

Which proportion of websites uses HTTP/2 today?

- **A:** less than 25%
- **B: 25%–50%**
- **C:** 50%–75%
- **D:** over 75%



# Modern HTTP versions

- New version: **HTTP/2** (originally **SPDY** by Google)
  - **Documented** protocol: RFC 7540 (96 pages, 2015)
  - Used by **47%** of websites today<sup>1</sup>

---

<sup>1</sup><https://w3techs.com/technologies/details/ce-http2/all/all>, November 2021



# Modern HTTP versions

- New version: **HTTP/2** (originally **SPDY** by Google)
  - **Documented** protocol: RFC 7540 (96 pages, 2015)
  - Used by **47%** of websites today<sup>1</sup>
- Development version: **HTTP/3** (November 2018) from a Google project (QUIC) to accelerate **TCP**
  - Experimental support in Chrome and Firefox
  - Used by **24%** of websites today (November 2021)

---

<sup>1</sup><https://w3techs.com/technologies/details/ce-http2/all/all>, November 2021

# HTTP queries (1.1)

- From **client** to **server**, TCP connection (+TLS)

```
GET /wiki/Telecom_Paris HTTP/1.1
```

```
Host: en.wikipedia.org
```

→ [http://en.wikipedia.org/wiki/Telecom\\_Paris](http://en.wikipedia.org/wiki/Telecom_Paris)

**Method** Several choices:

GET Most common

POST Forms, side effects

HEAD Only metadata

**others** PUT, DELETE...

**Path** That of the URL

**Version** Here, 1.1

**Headers** More info (cf. later)

**Body** Give some parameters (with POST)

# HTTP responses

- From **server** to client, in the same connection

```
HTTP/1.1 200 OK
```

```
Content-Type: text/html; charset=UTF-8
```

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    (...)
```

- **Status code** and **explanations**
- **Headers**
- **Response** (e.g., page content)

# Most common status codes

## **2XX** Success

- 200: OK

## **3XX** Redirection

- 301: permanent
- 302: temporary

## **4XX** Client error

- 400: syntax error
- 401: authentication required
- 403: forbidden
- 404: not found

## **5XX** Server error

- 500: internal server error

# Paths and parameters

- Paths are typically **hierarchical** (separator: /)
- Unix conventions: `https://en.wikipedia.org/./wiki/./`
- Can add **key-value** parameters
- **Example** : `https://www.google.com/search?q=telecom&ie=utf-8&oe=utf-8&client=iceweasel-a`
- **Percent-encoding** for special characters:  
`https://fr.wikipedia.org/wiki/T%C3%A9l%C3%A9com_Paris`

# Table of Contents

HTTP

Headers

Other HTTP notions

HTTP 1 vs HTTP 2

# Client Host header

- Indicate again the **original domain name**
- Find the correct **virtual host**

Host: en.wikipedia.org

## Other main client headers

- **User-Agent**: declare which browser is used

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:17.0)
```

```
Gecko/20130810 Firefox/17.0 Iceweasel/17.0.8
```



## Other main client headers

- **User-Agent**: declare which browser is used

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:17.0)
Gecko/20130810 Firefox/17.0 Iceweasel/17.0.8
```

- **Accept** and **Accept-\***: give preferred filetype and language

```
Accept: text/html,application/xhtml+xml,
application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
```

## Other main client headers

- **User-Agent**: declare which browser is used

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:17.0)
Gecko/20130810 Firefox/17.0 Iceweasel/17.0.8
```

- **Accept** and **Accept-\***: give preferred filetype and language

```
Accept: text/html,application/xhtml+xml,
application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
```

- **Referer**: declare the previous webpage

```
Referer: https://en.wikipedia.org/wiki/Telecom_Paris
```

## Other main client headers

- **User-Agent**: declare which browser is used

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:17.0)
Gecko/20130810 Firefox/17.0 Iceweasel/17.0.8
```

- **Accept** and **Accept-\***: give preferred filetype and language

```
Accept: text/html,application/xhtml+xml,
application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
```

- **Referer**: declare the previous webpage

```
Referer: https://en.wikipedia.org/wiki/Telecom_Paris
```

- **Range**: request only part of content (e.g., resume a download)

# Main server headers

- **Server**: declare the server software
- **Content-Type** and **Content-Length**: declare the file type, encoding, size (progress bar)

# Table of Contents

HTTP

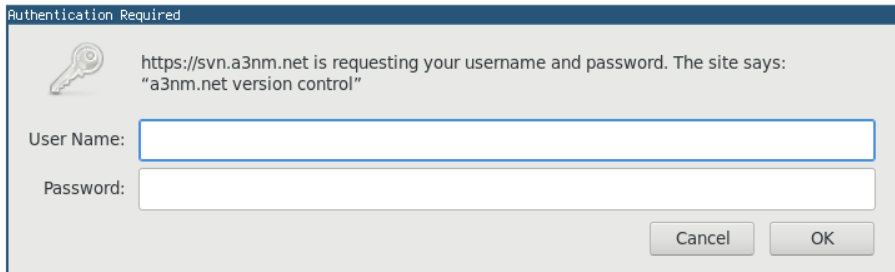
Headers

Other HTTP notions


HTTP 1 vs HTTP 2

# Basic and digest authentication

- HTTP can **authenticate** the client (cleartext/digest)



Authentication Required

 https://svn.a3nm.net is requesting your username and password. The site says:  
"a3nm.net version control"

User Name:

Password:

Cancel OK

- **Insecure** unless HTTPS is used
- Still **not very flexible** for websites

- **Proxy**: relay queries for someone else
- Main uses:
  - **Filter** or  **censor** content
  - **Log** the activity, keep a **cache**
  - **Anonymize** the query. Example: **Tor** anonymization network
- Difficult with **HTTPS** (the proxy no longer sees the content!)

# Content delivery networks (CDNs)

- Ensure that **static content** can be widely and reliably distributed
  - e.g., JSDelivr, BootstrapCDN, Cloudflare, Google Hosted Libraries, Google Fonts
- Often work together with **Internet Service Providers** (ISPs)
- Optimize the **connection** between the CDN datacenter and content provider
- Often provide **bot filtering**, **DDOS protection**, etc.
- Also: Facebook's **Instant Articles**, and **Google AMP**



# Caching

- **Save** the result of a query to avoid doing the query again
- Web browsers usually have a **cache** ; also proxies
- The server can indicate **whether** a response should be cached and **for how long**

**Cache-Control** Indicates whether to cache

**Expires** Expiry date

**ETag** Version identifier

- Client :

**If-Modified-Since** Get the content if modified since some date

**If-None-Match** Get the content if the ETag has changed

# Cookies

- No **sessions** in HTTP
- The server can ask the client to **store** a value:  
`Set-Cookie: name=value; option1; option2:`
  - **expires**: expiry date (can be in the distant future)
  - can limit the scope (domain, path), etc.
- The client will **provide the value** with every query:  
`Cookie: name=value`
- Of course the client can decide to **alter** cookies or **remove** them

# Using cookies

- Storing an opaque **session identifier**
- Ensuring that the user remains **logged in** for a long time
- **Privacy risk**: can track a user (hence: EU cookie consent)
- **Security risk**: with a stolen cookie, you can impersonate the user

# Table of Contents

HTTP

Headers

Other HTTP notions

HTTP 1 vs HTTP 2

# Compression

- With HTTP 1.1, compression is **possible** if both the client and server support it  
Accept-Encoding: gzip, deflate

# Compression

- With HTTP 1.1, compression is **possible** if both the client and server support it  
`Accept-Encoding: gzip, deflate`
- With HTTP 2, even **headers** can be compressed

## Connection type

- HTTP 1.0 used to **close** the connection after one query: inefficient!

## Connection type

- HTTP 1.0 used to **close** the connection after one query: inefficient!
- HTTP 1.1: the connection **stays open** by default (until timeout)  
Connection: keep-alive
- **Pipelining**: send multiple queries and get responses in order
  - Not commonly used because **badly supported** in practice



## Connection type

- HTTP 1.0 used to **close** the connection after one query: inefficient!
- HTTP 1.1: the connection **stays open** by default (until timeout)  
Connection: keep-alive
- **Pipelining**: send multiple queries and get responses in order
  - Not commonly used because **badly supported** in practice
- With HTTP 2 you can do **multiplexing**: send many queries and get responses in arbitrary order
- With HTTP 2, the server can also push resources to the client **before** it requests them

- Matériel de cours inspiré de notes par Pierre Senellart et Georges Gouriten
- Merci à Pierre Senellart pour sa relecture