NodeJS

środowisko i technologia ServerSide

PAWEŁ ŁUKASZUK

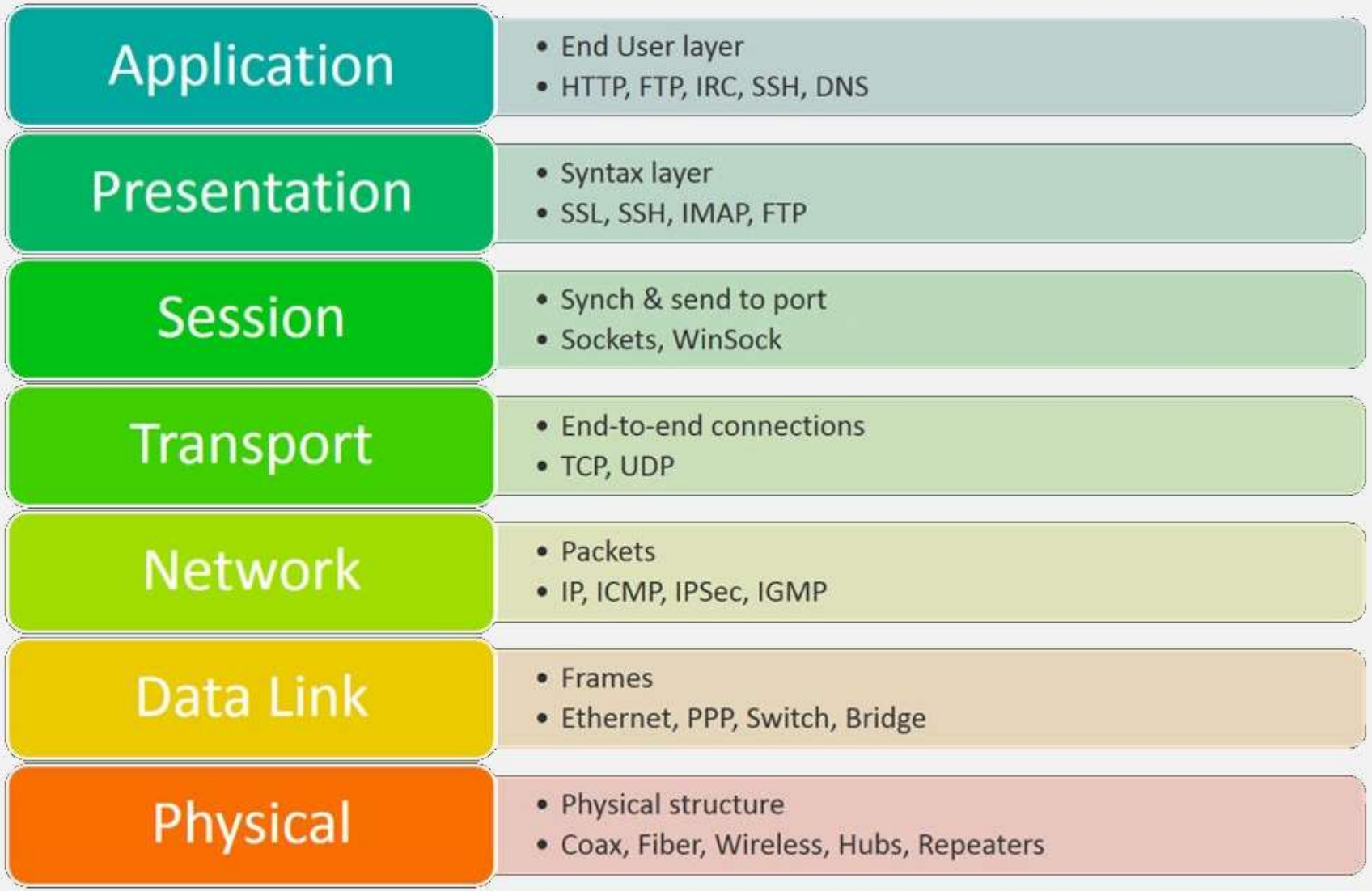
Protocol

Protocol is set of rules

A communication protocol is a group of guidelines that allows for efficient and effective data transfer…

when followed by all parties to the communication

OSI Model


Application
• End User layer
• HTTP, FTP, IRC, SSH, DNS
Presentation
• Syntax layer
• SSL, SSH, IMAP, FTP
Session
• Synch & send to port
• Sockets, WinSock
Transport
• End-to-end connections
• TCP, UDP
Network
• Packets
• IP, ICMP, IPSec, IGMP
Data Link
• Frames
• Ethernet, PPP, Switch, Bridge
Physical
• Physical structure
• Coax, Fiber, Wireless, Hubs, Repeaters

Communication protocols

Types of protocols:

By connection type:
- connection-oriented
- connectionless

By reliability:
- reliable
- unreliable

Transmission Control Protocol

TCP - Transmission Control Protocol – connection-based, reliable, communication protocol used to exchange data between processes running on different machines.

Over 85% of total Internet traffic is TCP traffic.

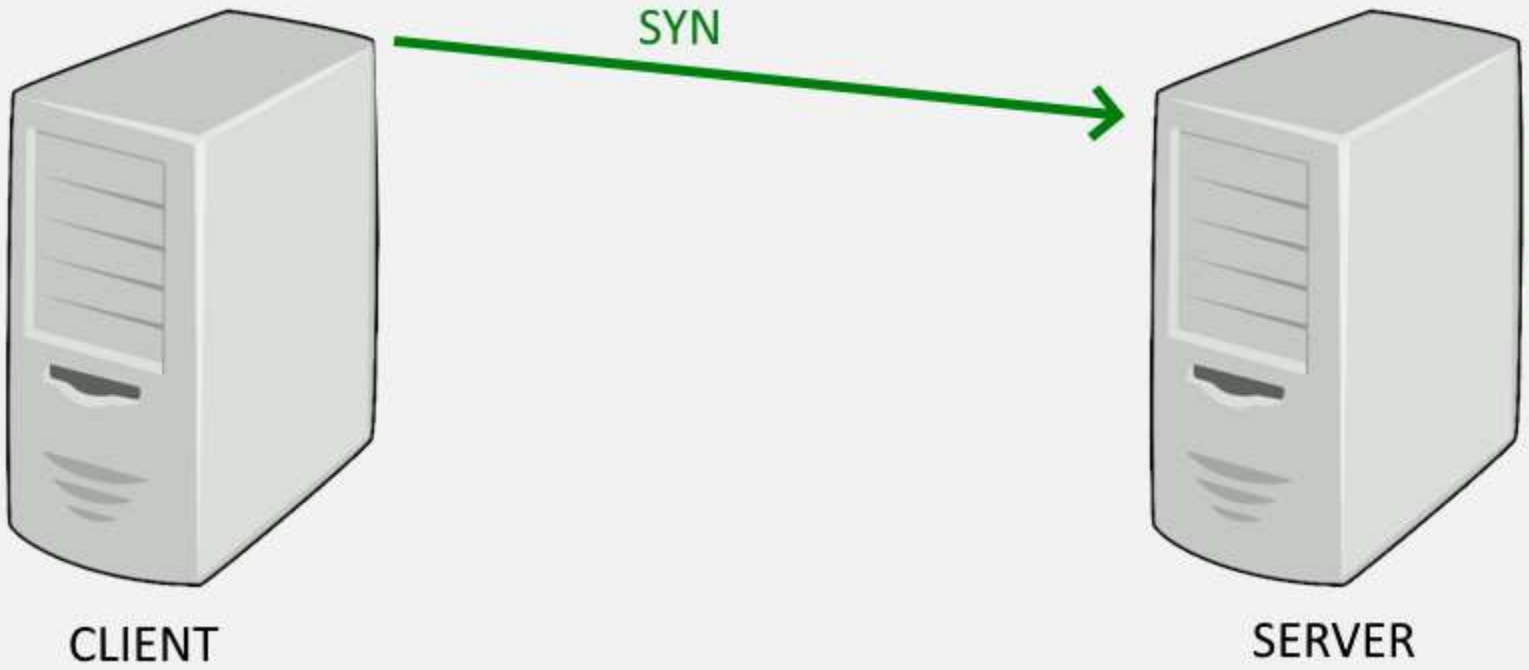
Connection establishment

CLIENT

SERVER

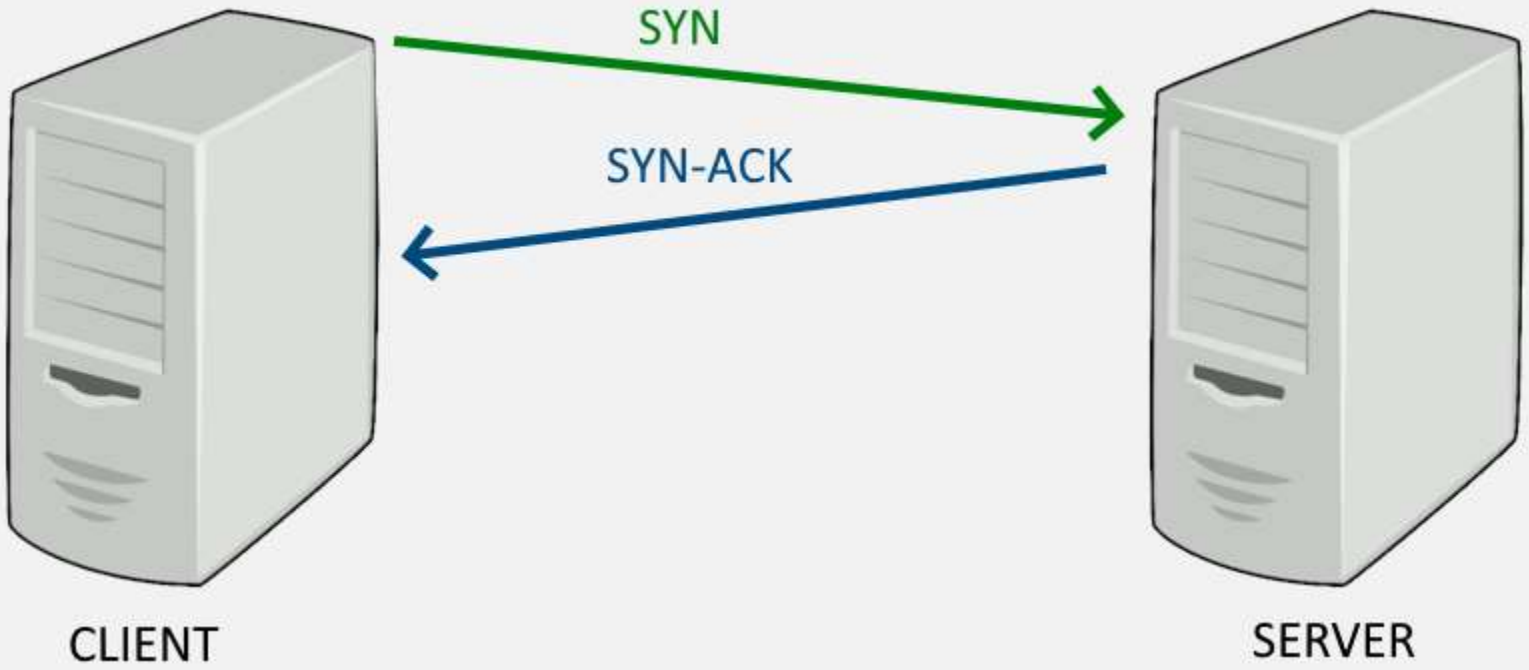
Connection establishment

Synchronize



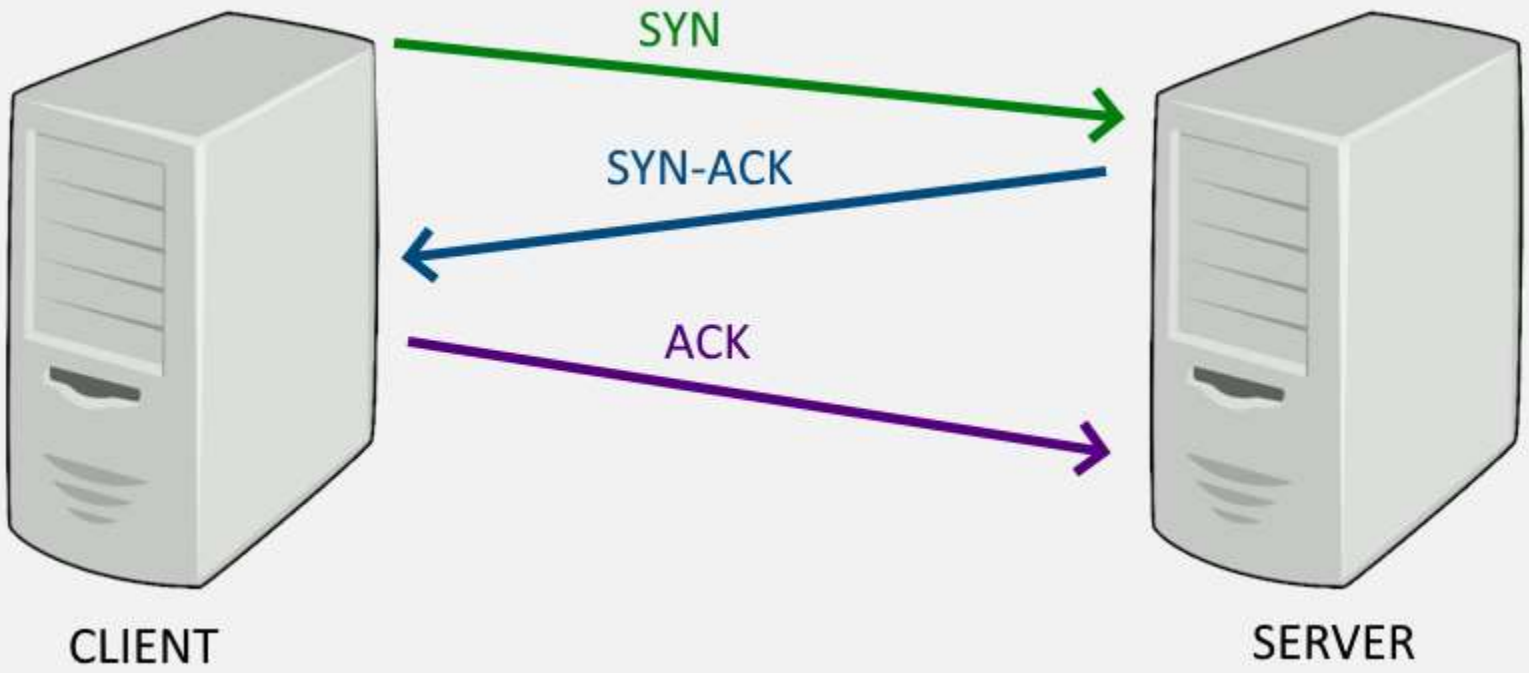
Connection establishment

Synchronize + Acknowledgement



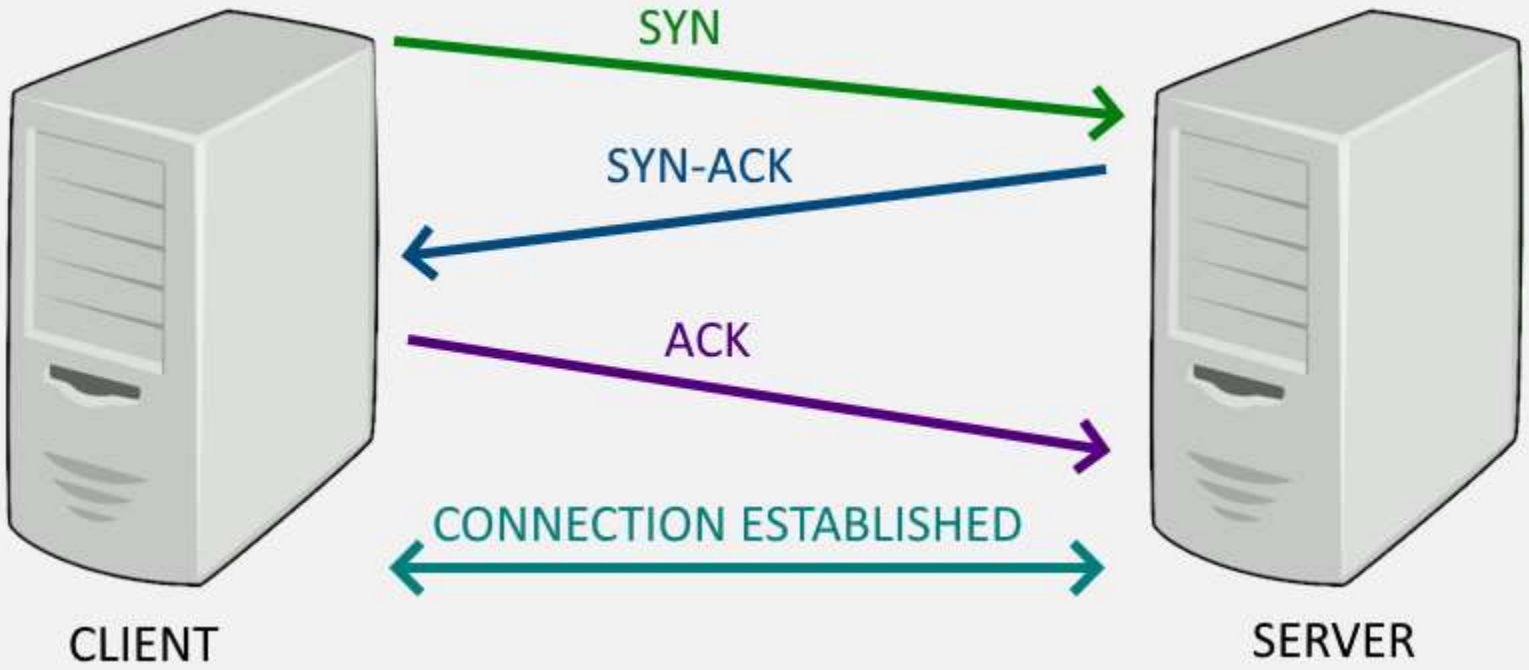
Connection establishment

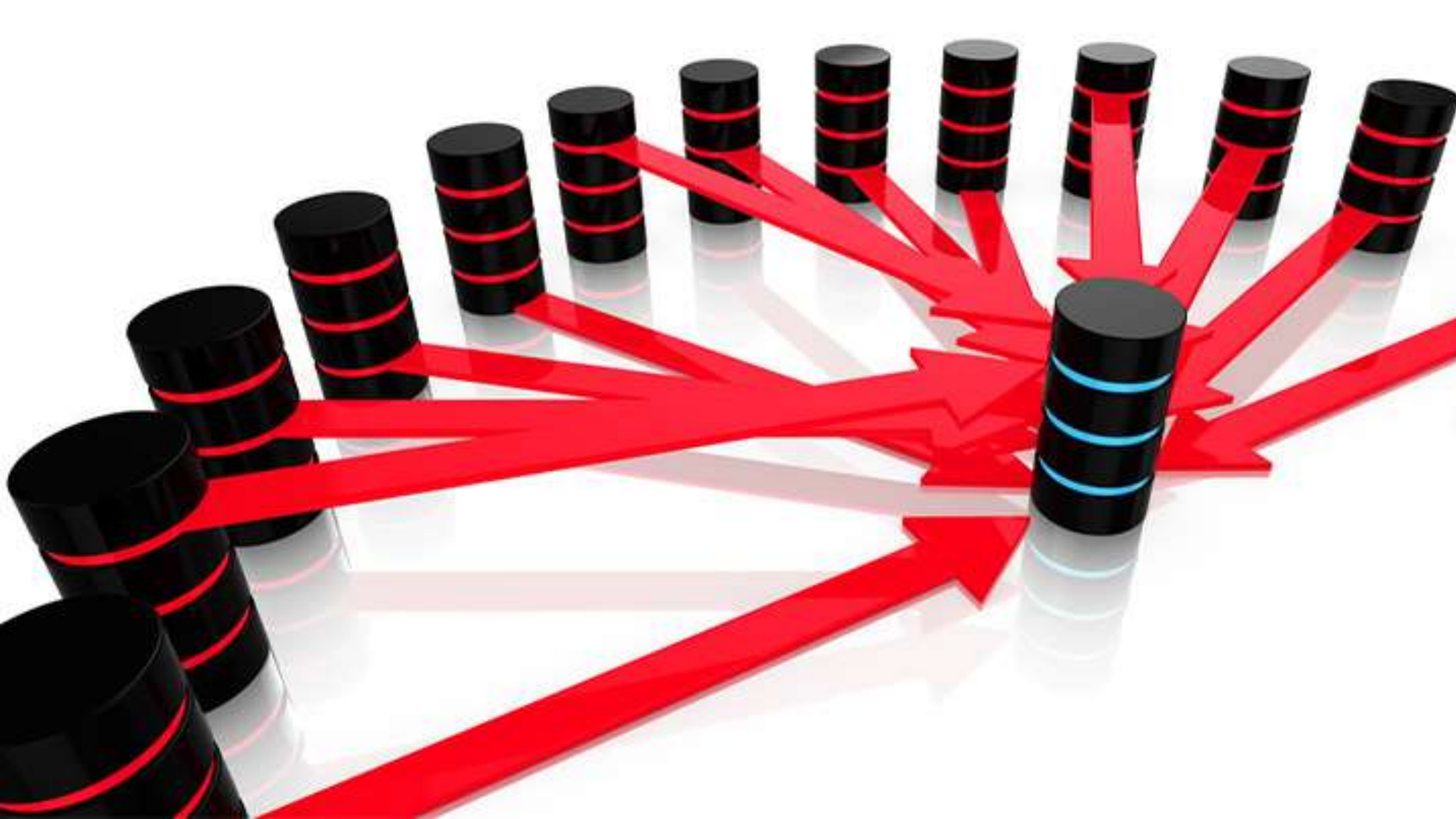
Acknowledgement



Connection establishment

Three-way handshake

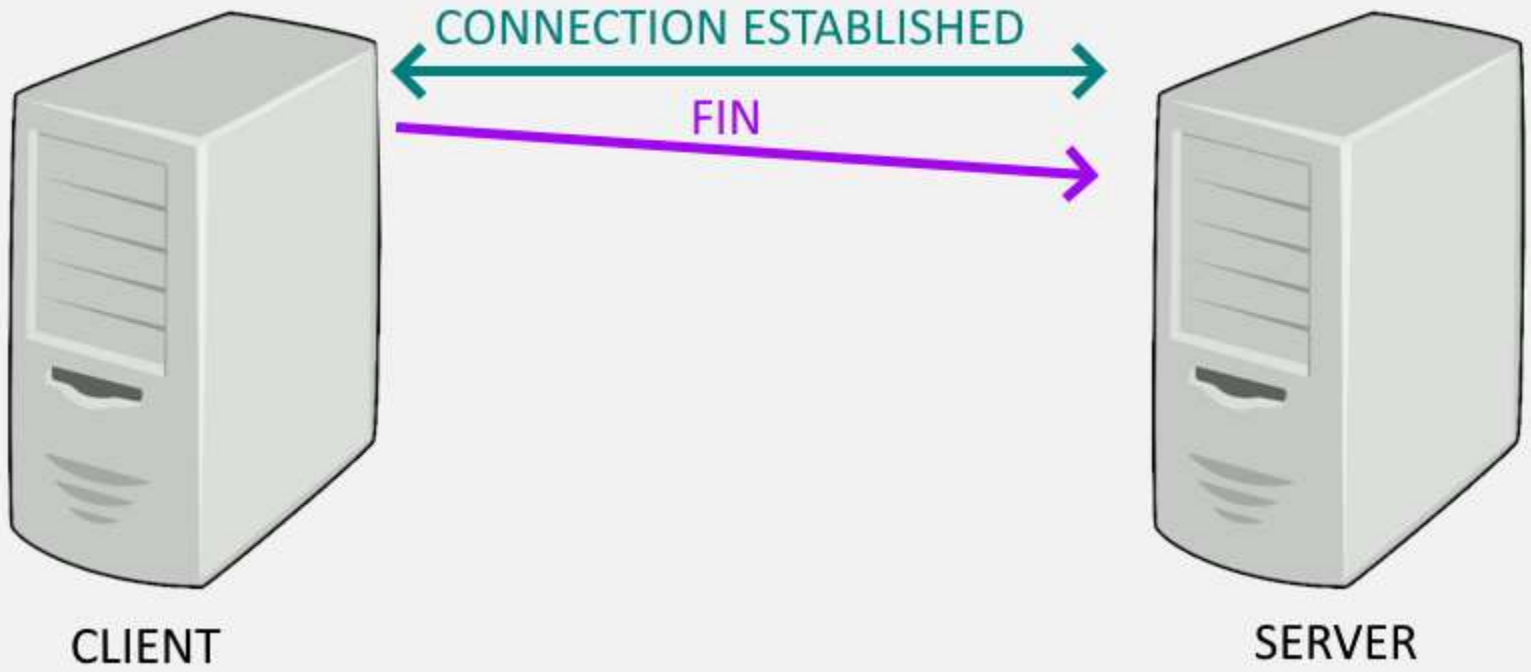




Connection termination

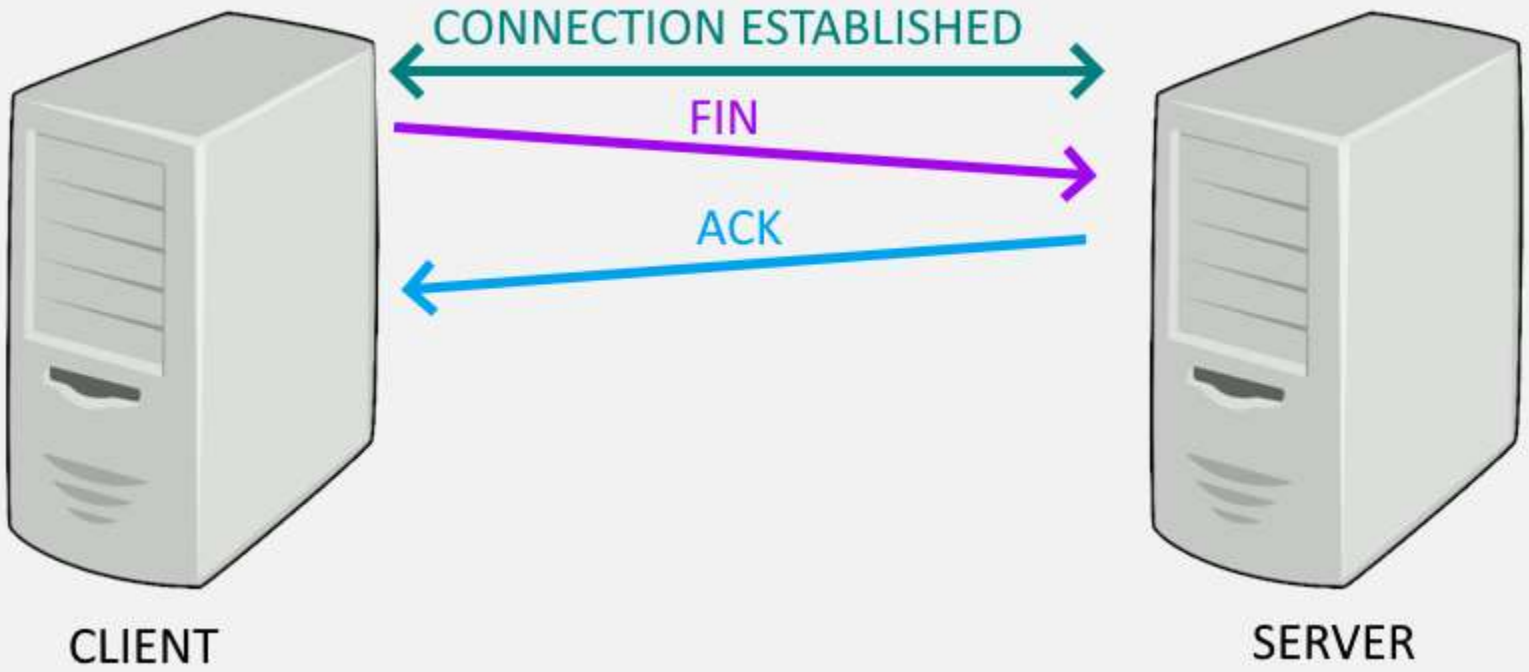
Connection termination

Finished



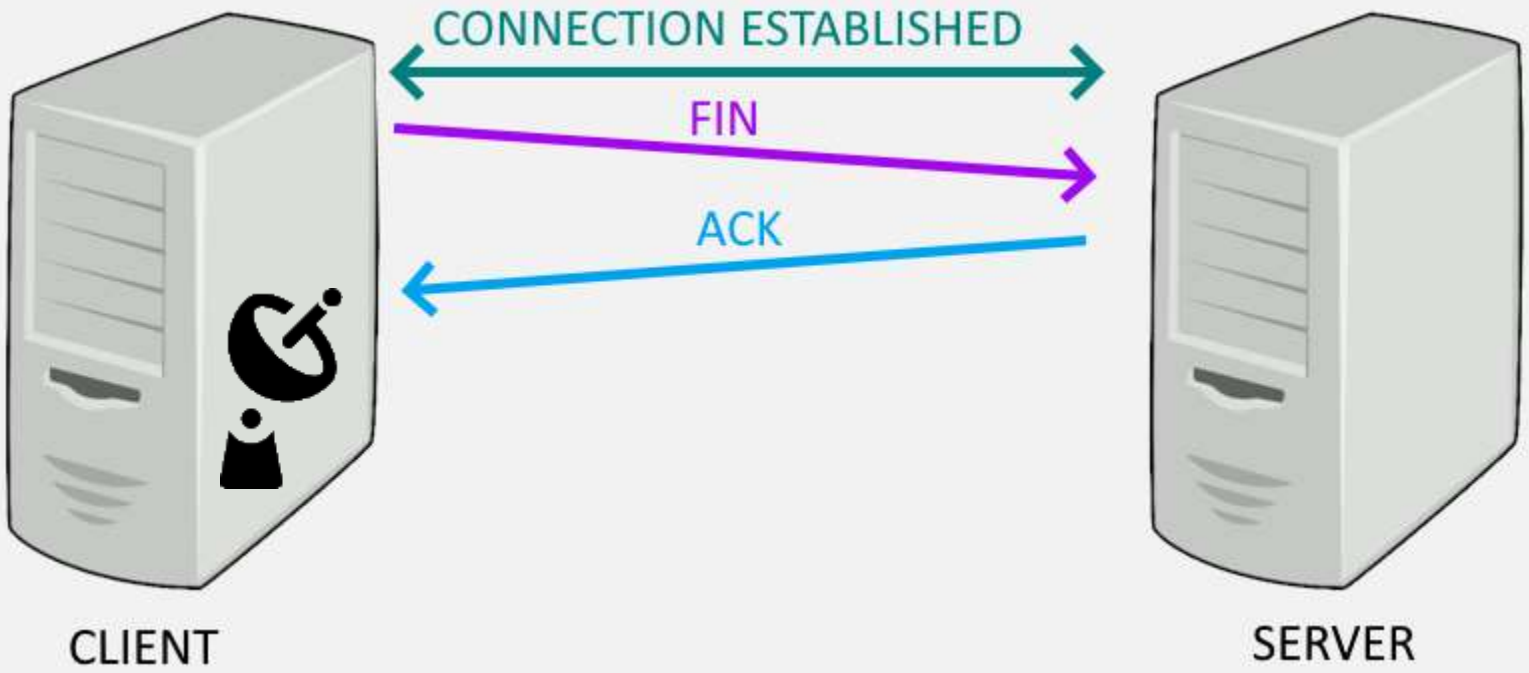
Connection termination

Acknowledgement



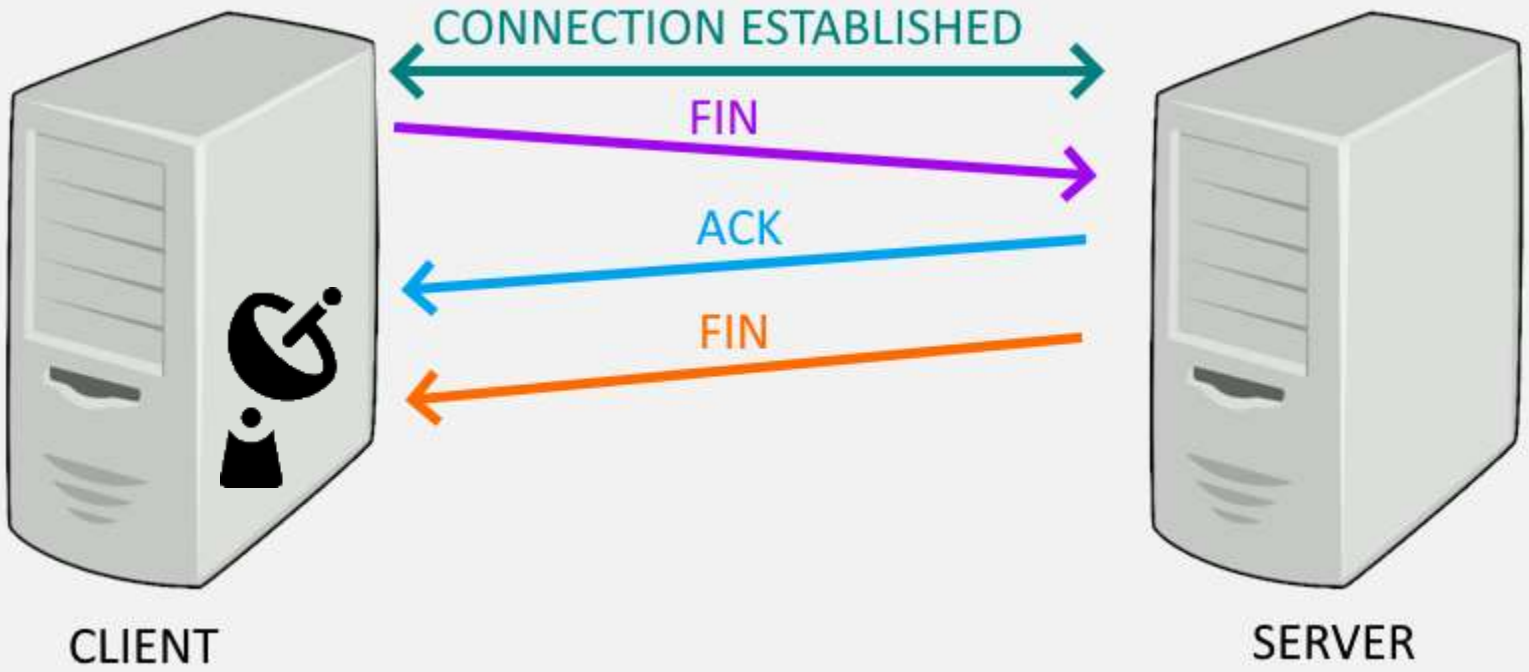
Connection termination

Acknowledgement



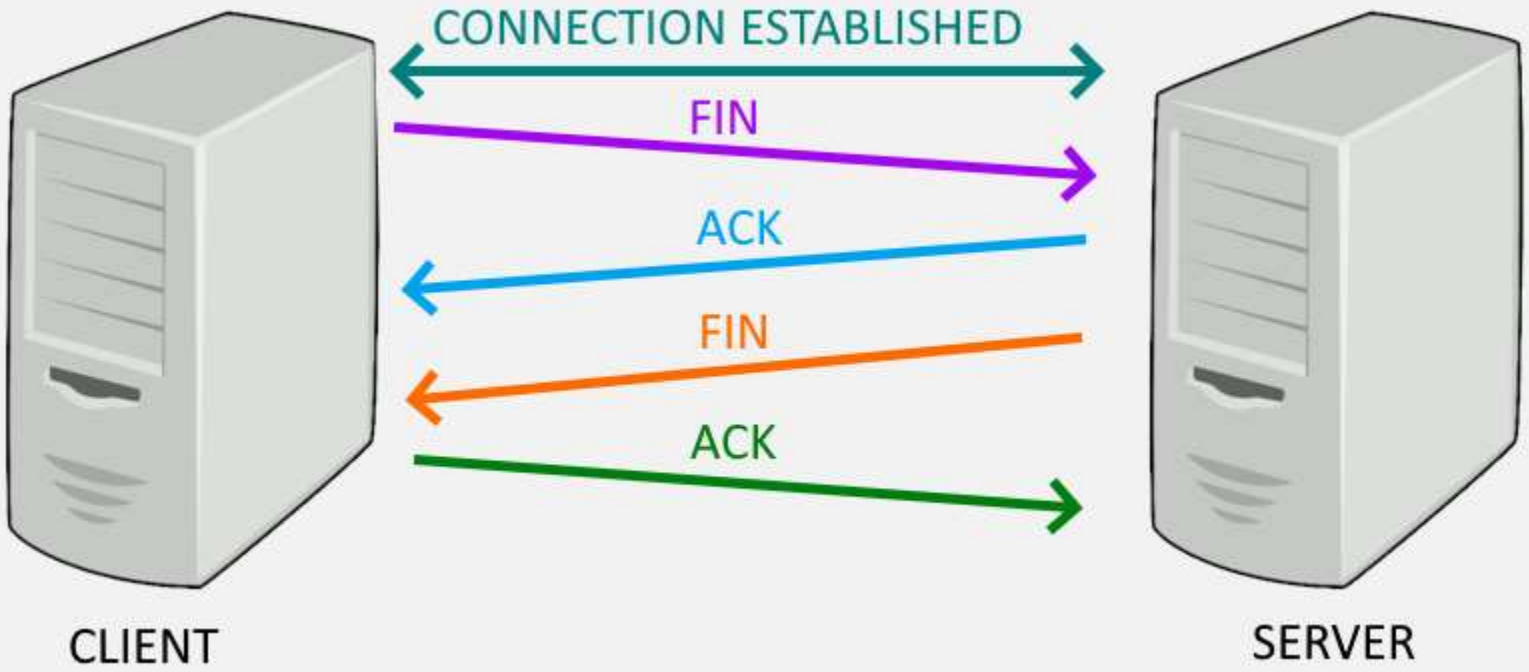
Connection termination

Finished



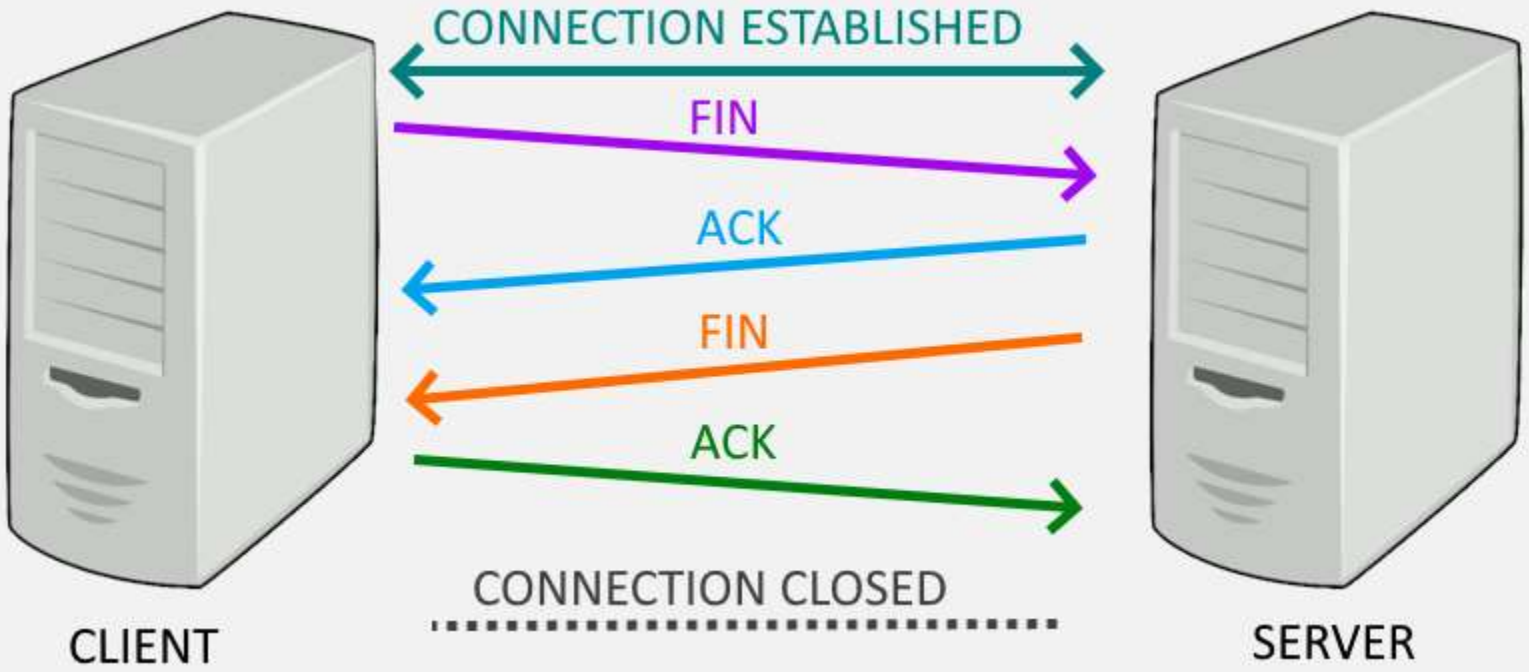
Connection termination

Acknowledgement



Connection termination

four-way handshakes or pair of two-way handshakes



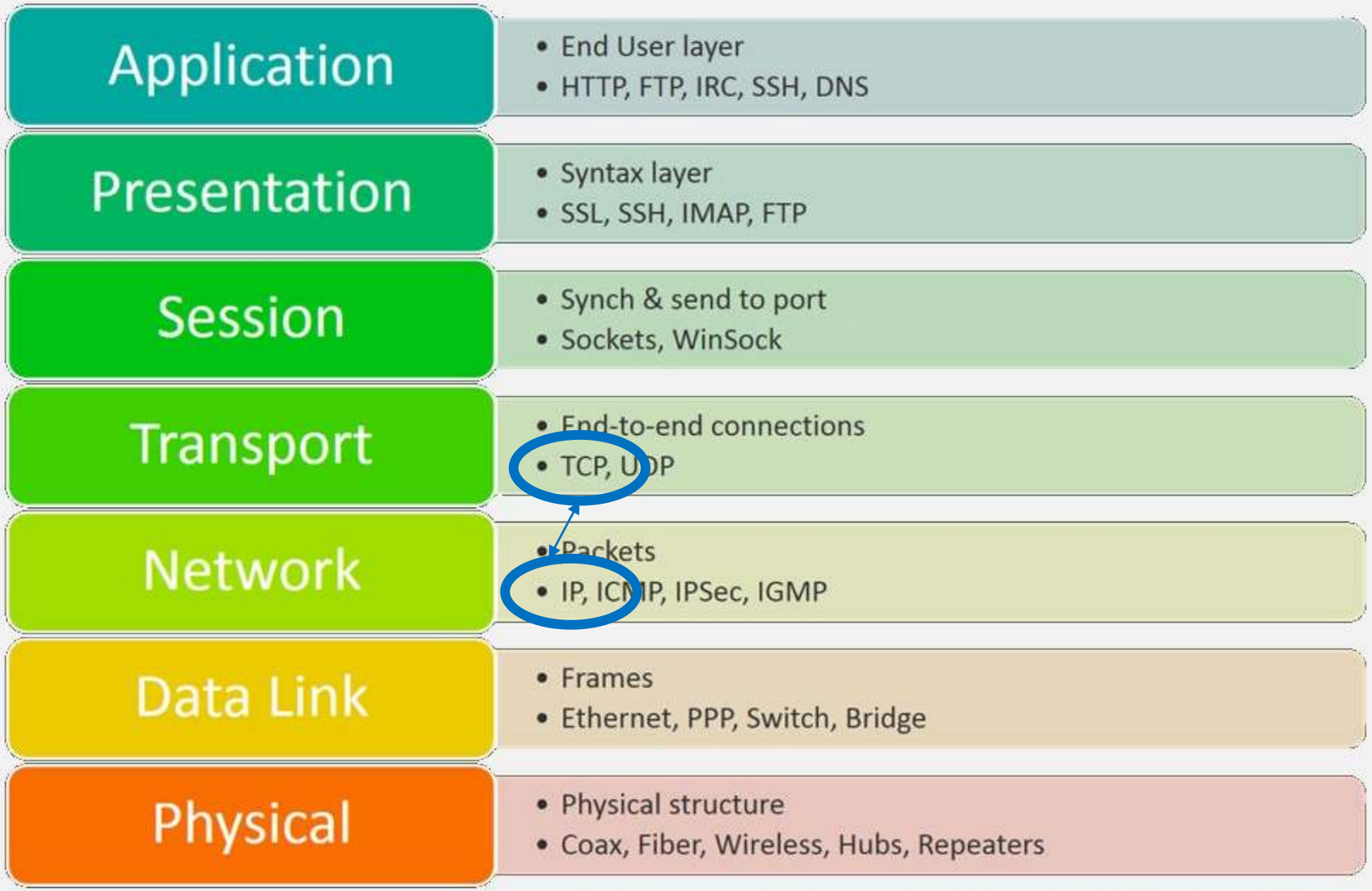
TCP/IP

The term TCIP/IP protocol stack is commonly used to refer to the Internet protocol suite since the TCP protocol is almost always based on the Internet protocol (IP).

Internet Protocol was designed explicitly as addressing protocol. The IP addresses in packets help in routing them through different nodes in a network until it reaches the destination system.

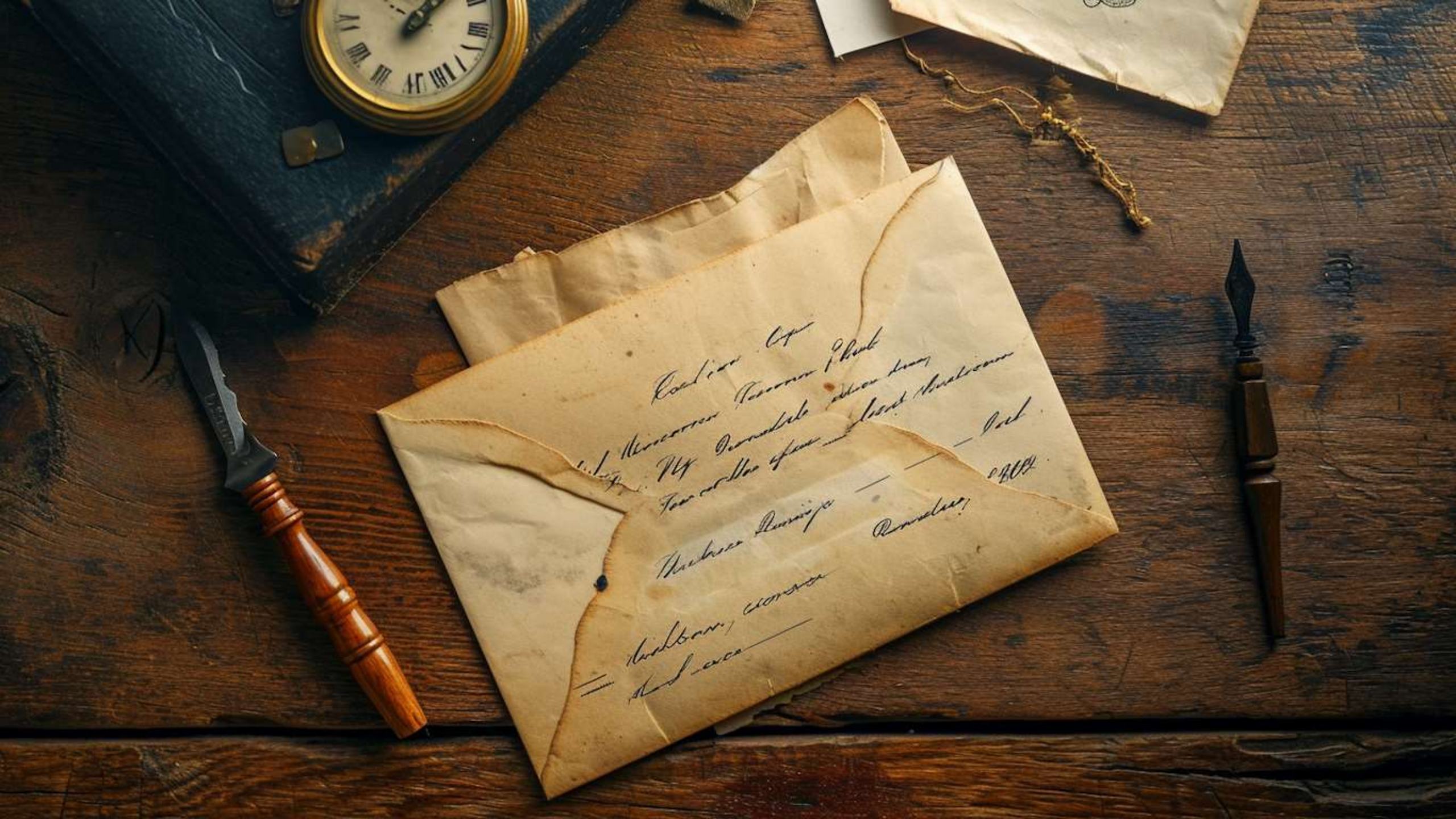
This connection is the foundation for the majority of public and local networks and network services.

OSI Model


Application
• End User layer
• HTTP, FTP, IRC, SSH, DNS
Presentation
• Syntax layer
• SSL, SSH, IMAP, FTP
Session
• Synch & send to port
• Sockets, WinSock
Transport
• End-to-end connections
• TCP, UDP
Network
• Packets
• IP, ICMP, IPSec, IGMP
Data Link
• Frames
• Ethernet, PPP, Switch, Bridge
Physical
• Physical structure
• Coax, Fiber, Wireless, Hubs, Repeaters

Other protocols

- User Datagram Protocol (UDP): substitute communication protocol to TCP implemented primarily for creating loss-tolerating and low-latency linking between different applications.
- Post office Protocol (POP): POP3 is designed for receiving incoming emails.
- Simple mail transport Protocol (SMTP): designed to send and distribute outgoing E-Mail.
- File Transfer Protocol (FTP): allows users to transfer files from one machine to another. Types of files may include program files, multimedia files, text files, and documents, etc.
- Hyper Text Transfer Protocol (HTTP): designed for transferring a hypertext among two or more systems.
- Hyper Text Transfer Protocol Secure (HTTPS): standard protocol to secure the communication among two computers one using the browser and other fetching data from web server.



URI, URL, URN

Uniform Resource Identifier - an Internet standard for identifying resources on the Web.

Uniform Resource Locator - an Internet standard for addressing resources on the Web.

Uniform Resource Name - globally unique persistent identifiers assigned within defined namespaces so they will be available for a long period of time, even after the resource which they identify ceases to exist or becomes unavailable

URI, URL, URN

URI – a resource identifier is specified but there is no information how to access it

example.org/absolute/URI/with/absolute/path/to/resource.txt

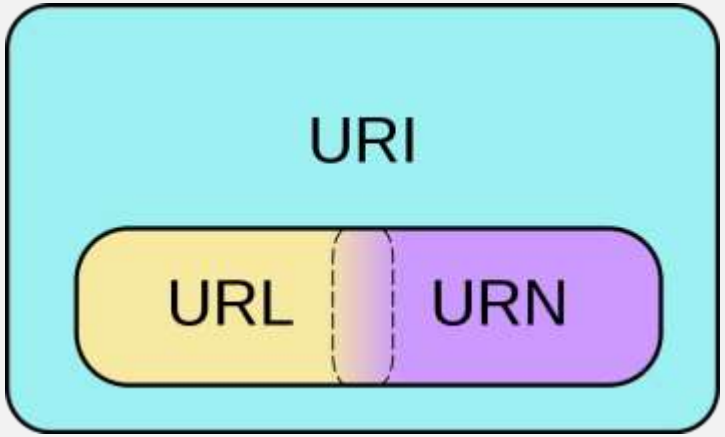
URL – defines the identifier of the resource and where and how to retrieve it (http server, HTTPS protocol)

https://example.org/absolute/URI/with/absolute/path/to/resource.txt

URN – namespace identifier and resource identifier

ISBN:951-0-18435-7 (ISBN-10)

ISSN:0167-6423



URL – common example

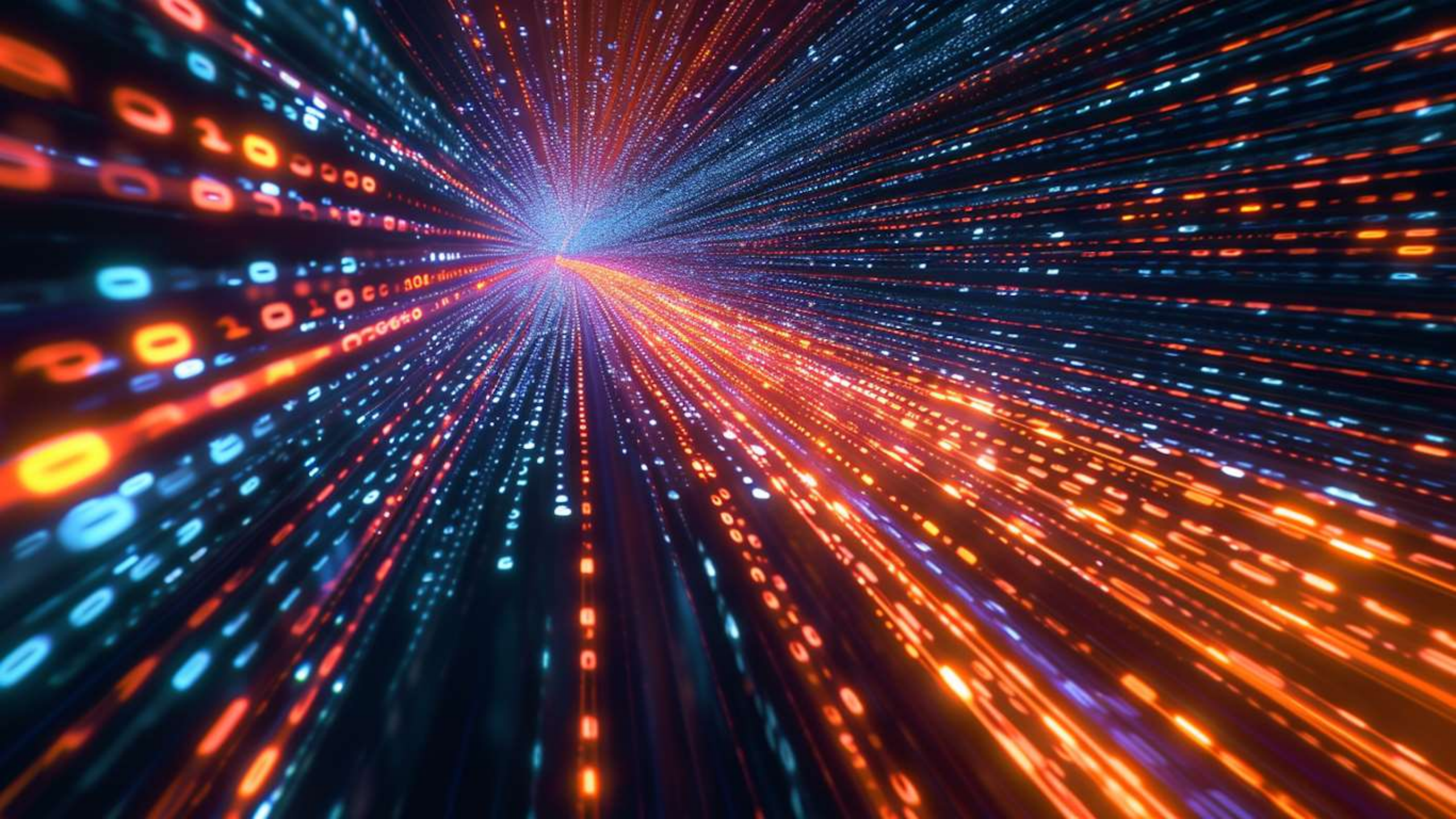
http://www.example.org/dir1/dir2/file.html?parametr1=value1&parametr2=value2#document_part

- http – protocol
- www.example.org – host (server address)
- dir1/dir2/file.html – resource
- parametr1=value1&parametr2=value2 – query parameters / query string
- #document_part – document part

URL – theoretical example

http://john:pswd@www.example.org:8080/dir1/dir2/file.html?parametr1=value1&parametr2=value2#document_part

- http – protocol
- john – login
- pswd - password
- www.example.org – host (server address)
- 8080 – port (for HTTP default is 80, for HTTPS: 443)
- dir1/dir2/file.html – resource
- parametr1=value1&parametr2=value2 – query parameters / query string
- #document_part – document part

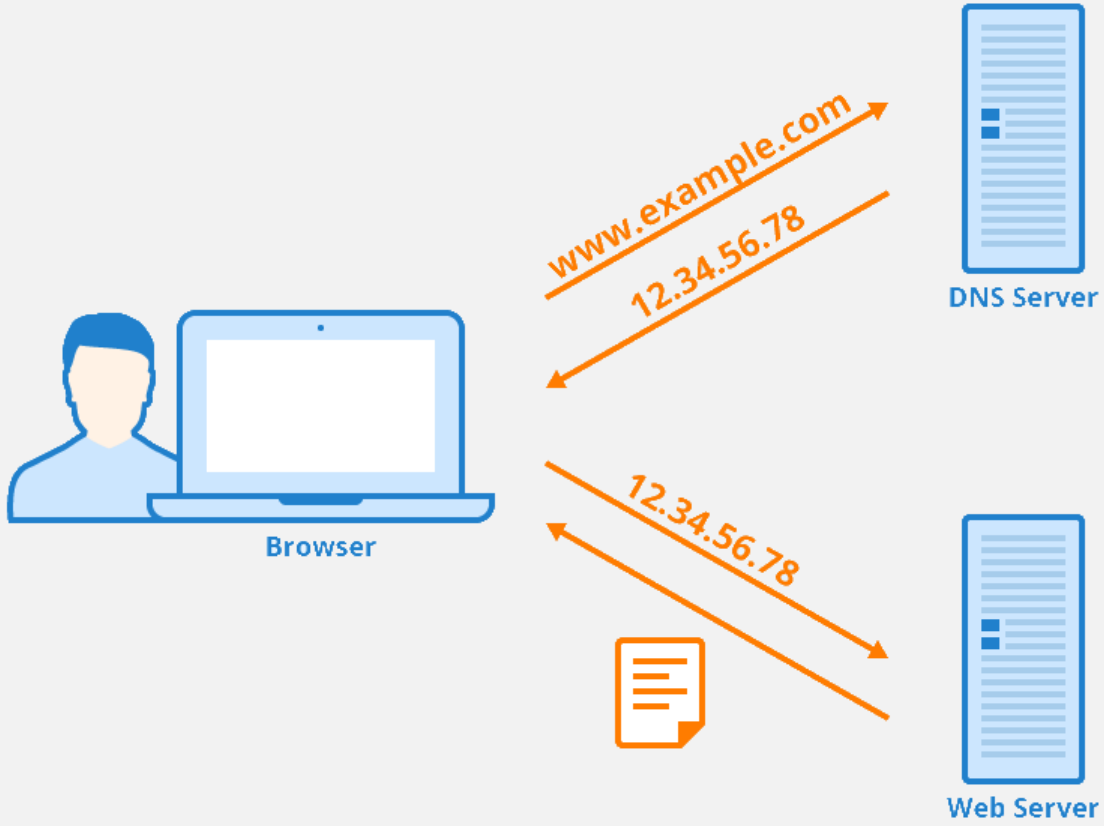


Domain Name System

DNS is the phonebook of the Internet.

The process of DNS resolution involves converting a hostname (such as www.example.com) into a computer-friendly IP address (such as 192.168.1.1).

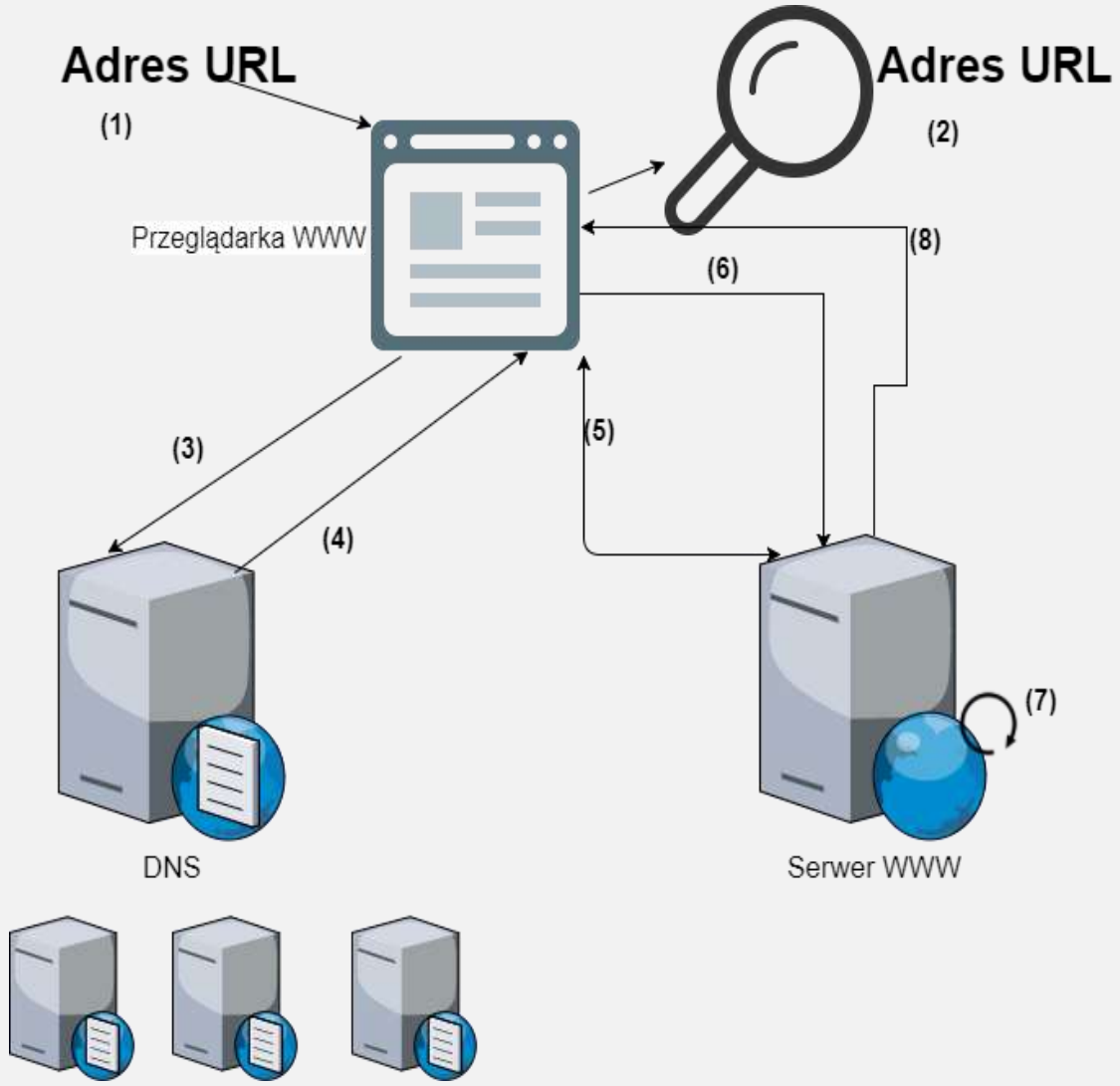
An IP address is given to each device on the Internet, and that address is necessary to find the appropriate Internet device - like a street address is used to find a particular home. When a user wants to load a webpage, a translation must occur between what a user types into their web browser (example.com) and the machine-friendly address necessary to locate the example.com webpage.



How web works

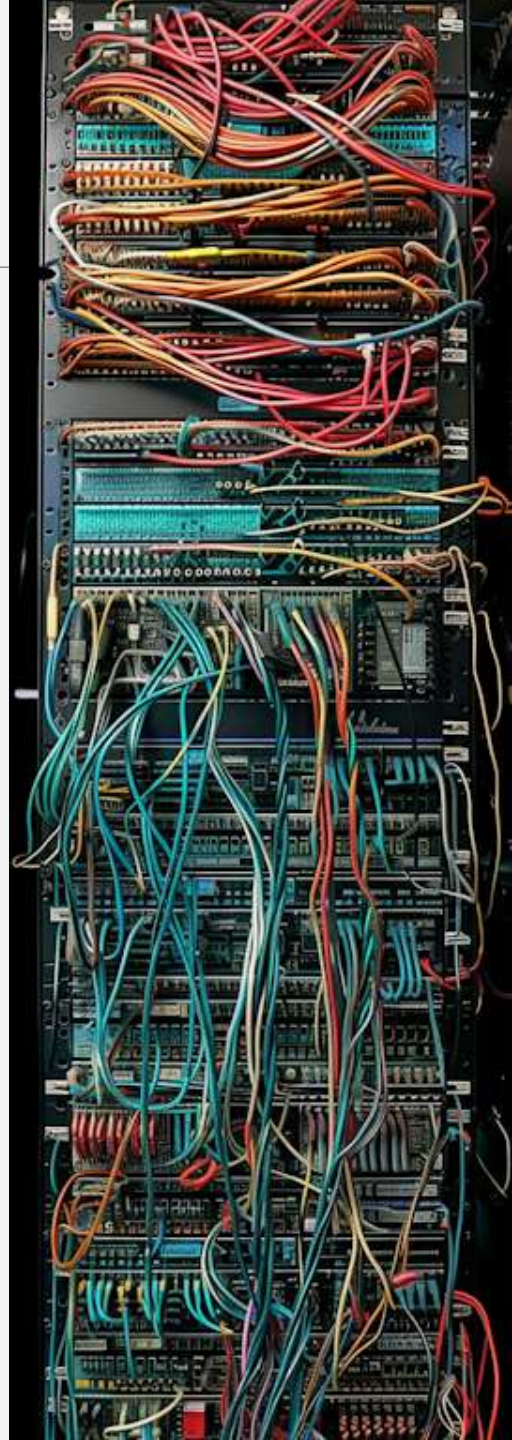
Steps to display a webpage:

1. user types URL in the browser
2. browser processes it
3. queries the IP address
4. the DNS server returns the IP
5. browser establishes a TCP connection with the web server
6. **browser sends a request**
7. **web server processes the request**
8. **web server sends response**
9. **browser displays response**



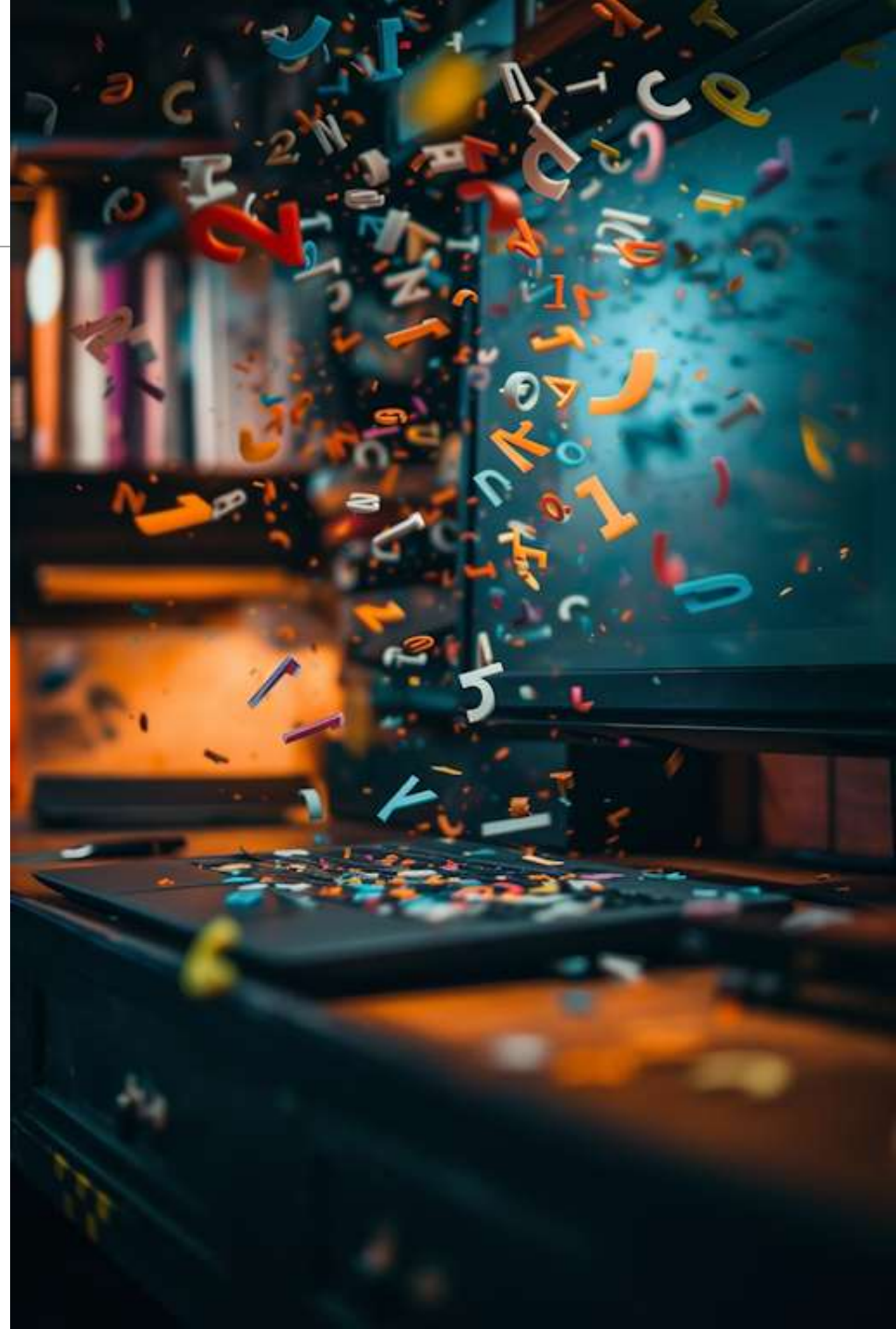
Important names and tools

- localhost - hostname that refers to the current device
- 127.0.0.1 - IP address of localhost
- > ping – command tool to check connection
  - ping google.com
  - ping 91.198.174.192
- > tracert – traceroute, command tool to trace TCP connection
  - tracert google.com
  - tracert 91.198.174.192
- > ipconfig – command tool in Windows to display network interface configuration
  - ipconfig /displaydns
  - ipconfig /flushdns
- C:\Windows\System32\drivers\etc\hosts
  – file used to override hostnames to IP mapping (hidden file!)



HTTP

- HyperText Transfer Protocol – a protocol for transferring hypertext documents
- provides a standardized way for client and server to communicate and exchange data
- determines the form of client requests for data and the form of the server's response to the request
- is used to transfer various types of data, such as images, videos, documents, etc.
- is based on TCP (Transmission Control Protocol)
- by default uses port 80



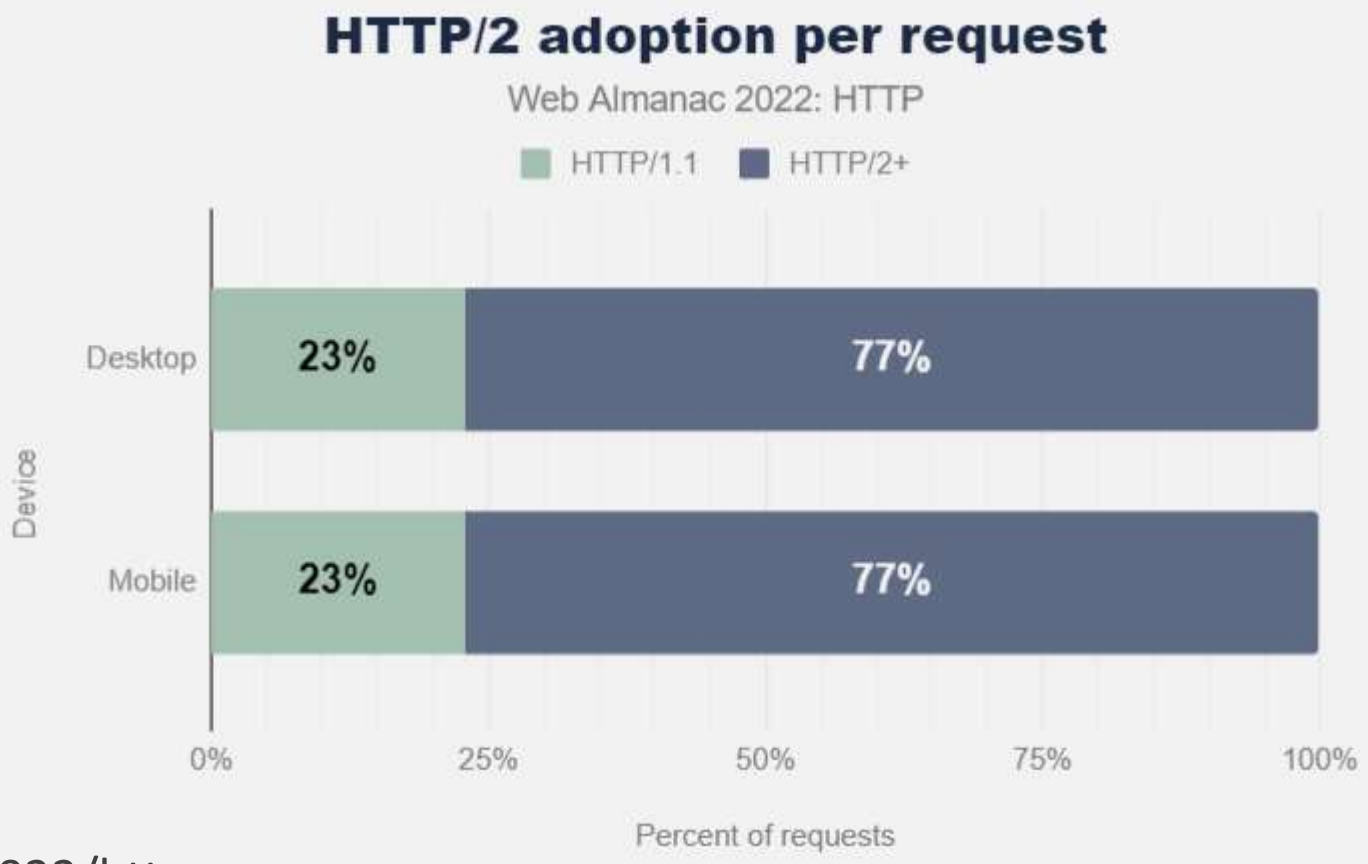
HTTP

- HTTP is connectionless
- HTTP is stateless
- HTTP delivers all types of data

- HTTP is insecure (issue resolved by HTTPS)

HTTP versions

- 1991 - 0.9
- 1996 - 1.0
- 1997 - 1.1
- 2015 - 2.0
- 2022 - 3.0



https://almanac.httparchive.org/en/2022/http

HTTP

- RFC-2616 (HTTP/1.1)
- RFC-7540 (HTTP/2)
- RFC-9114 (HTTP/3)

From the developer point of view:

- resources
- methods
- headers
- response codes
- authentication
- redirections

HTTP resources

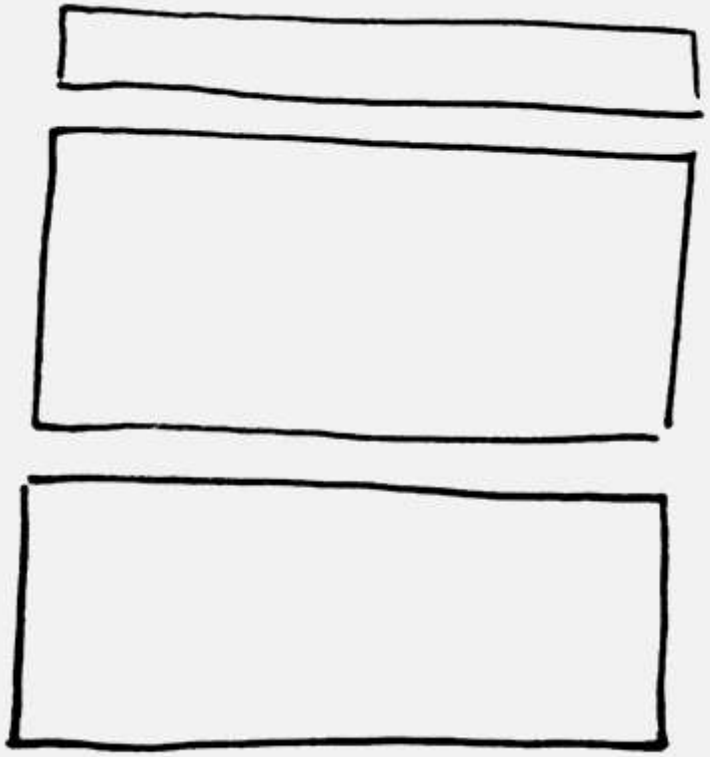
The target of an HTTP request is called a "resource", whose nature isn't defined further.

It can be a document, a photo, or anything else.
Each resource is identified by a Uniform Resource Identifier (URI) used throughout HTTP for identifying resources.

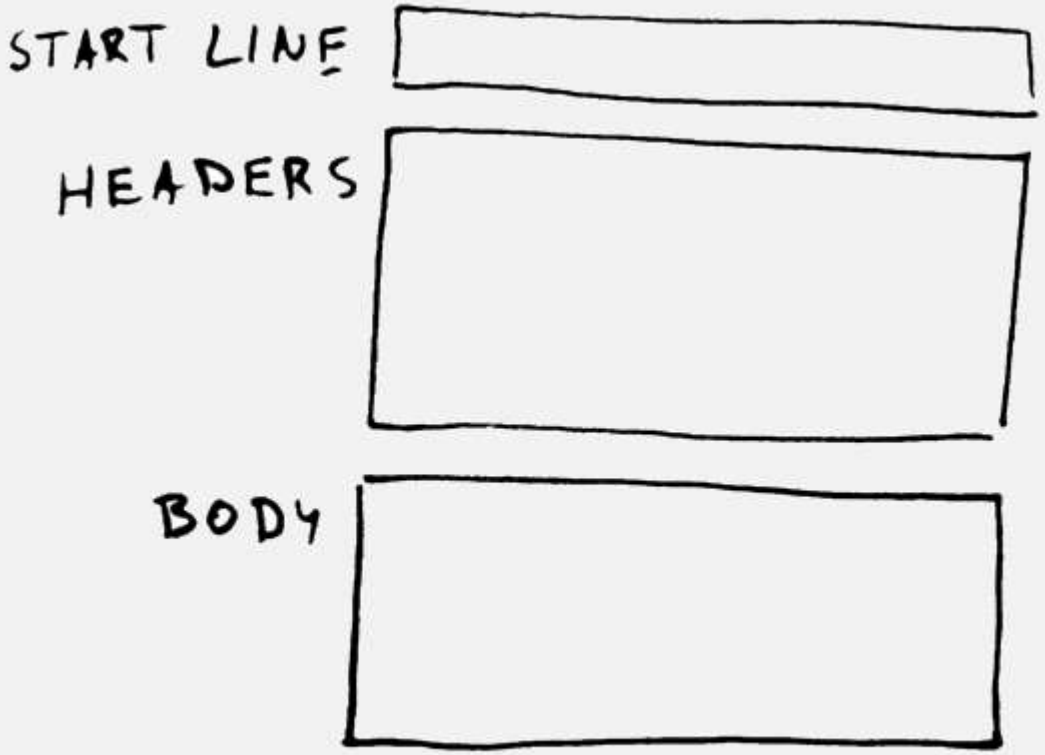
HTTP messages

- Request HTTP message
- Response HTTP message

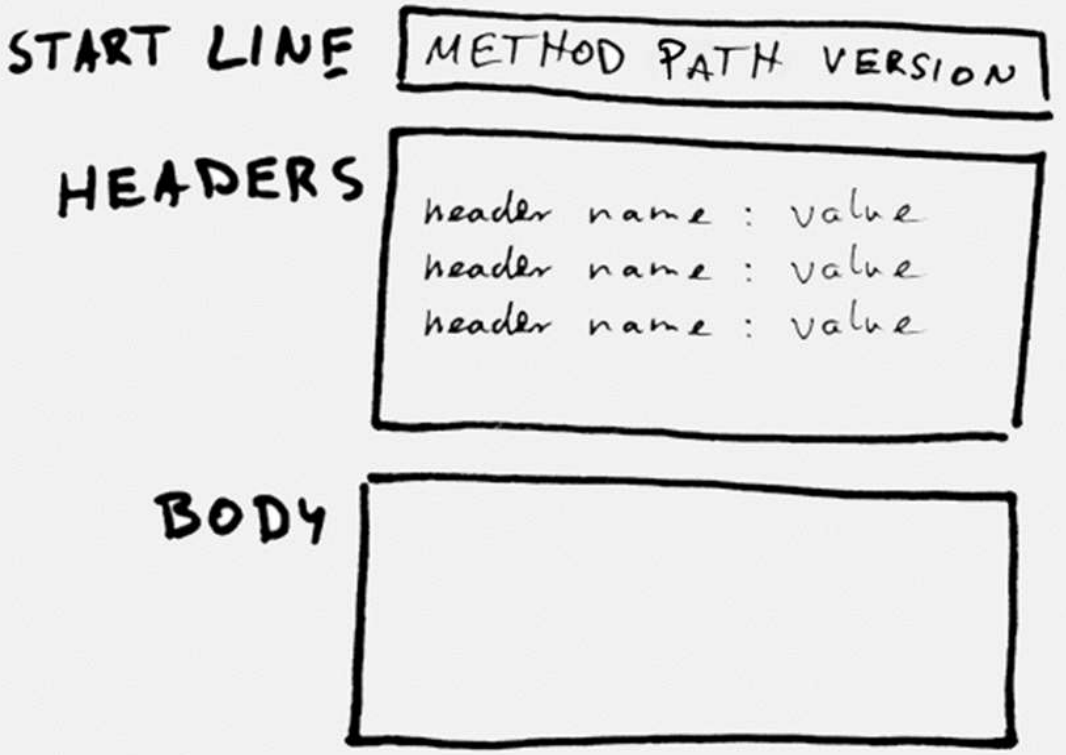
HTTP message



HTTP message



HTTP request message



HTTP request message

POST /test HTTP/1.1

START LINE

Host: example.com

Content-Type: application/json

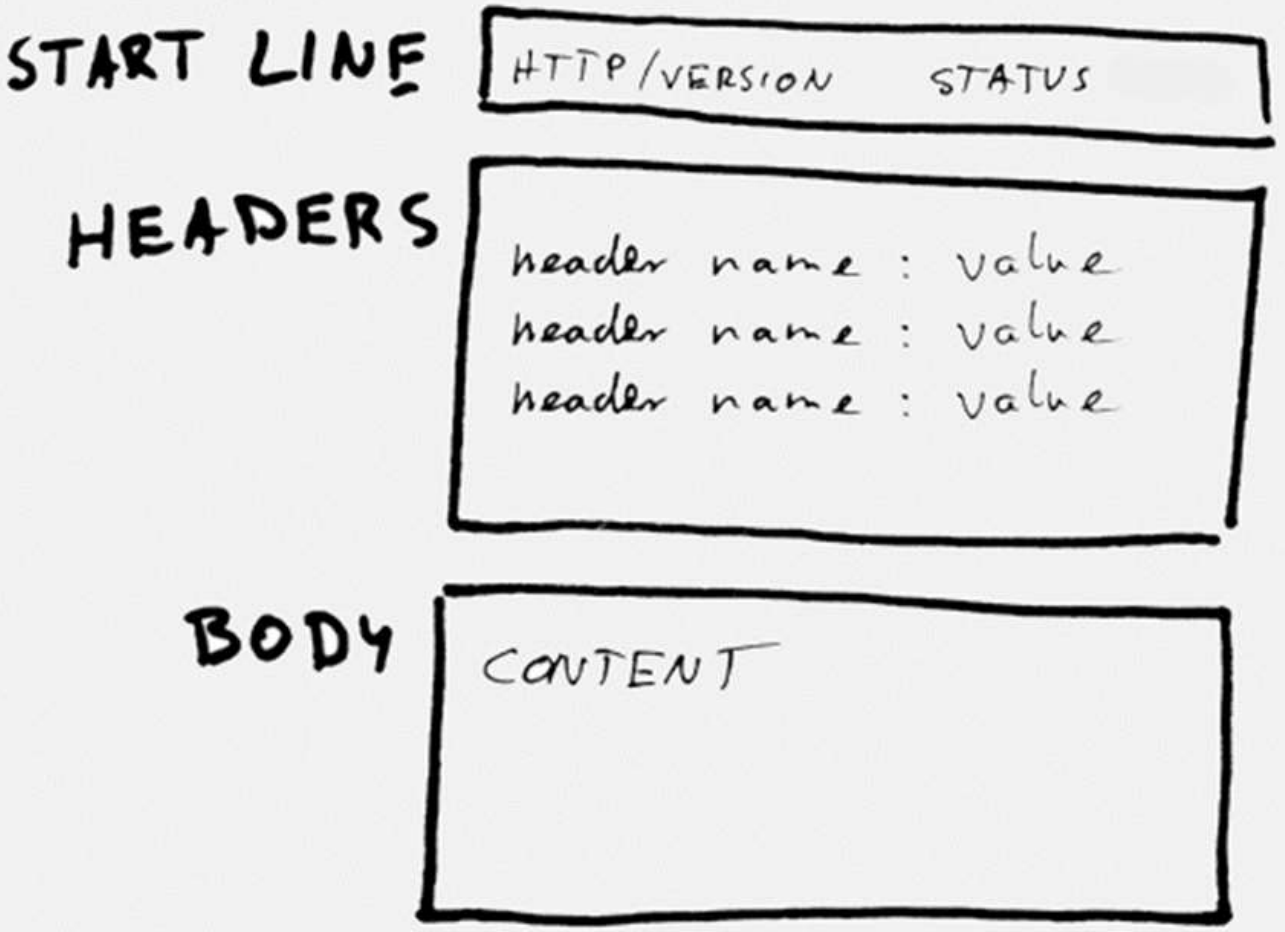
Content-Length: 111

HEADERS

```
{
   "start": 0, "count": 200,
   "sort": "NEWS_DATEADD_DESC",
   "yearMin": null, "yearMax": null
}
```

BODY

HTTP response message



HTTP response message

HTTP/1.1 200 OK

START LINE

Cache-Control: no-cache

Content-Type: application/json; charset=utf-8

Content-Length: 292

Date: Sat, 10 Feb 2024 09:31:48 GMT

HEADERS

{ ... user data }

BODY

HTTP methods

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource.

Each of them implements a different semantic.

Some common features are shared by a group of them: e.g. a request method can be safe, idempotent, or cacheable.

GET

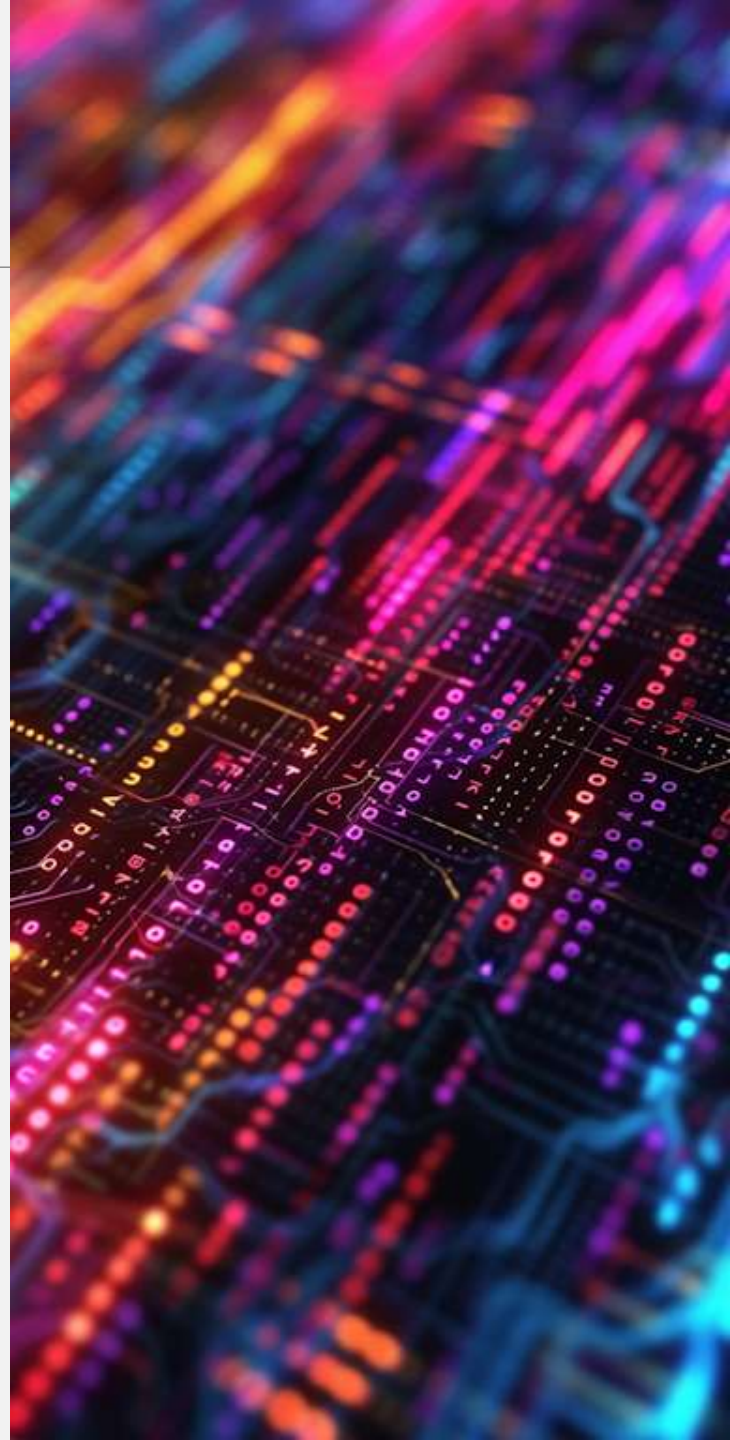
The GET method is used to request data from the specified resource.

Requests using the GET method should only retrieve the data.

Examples:

GET /users

GET /users/1



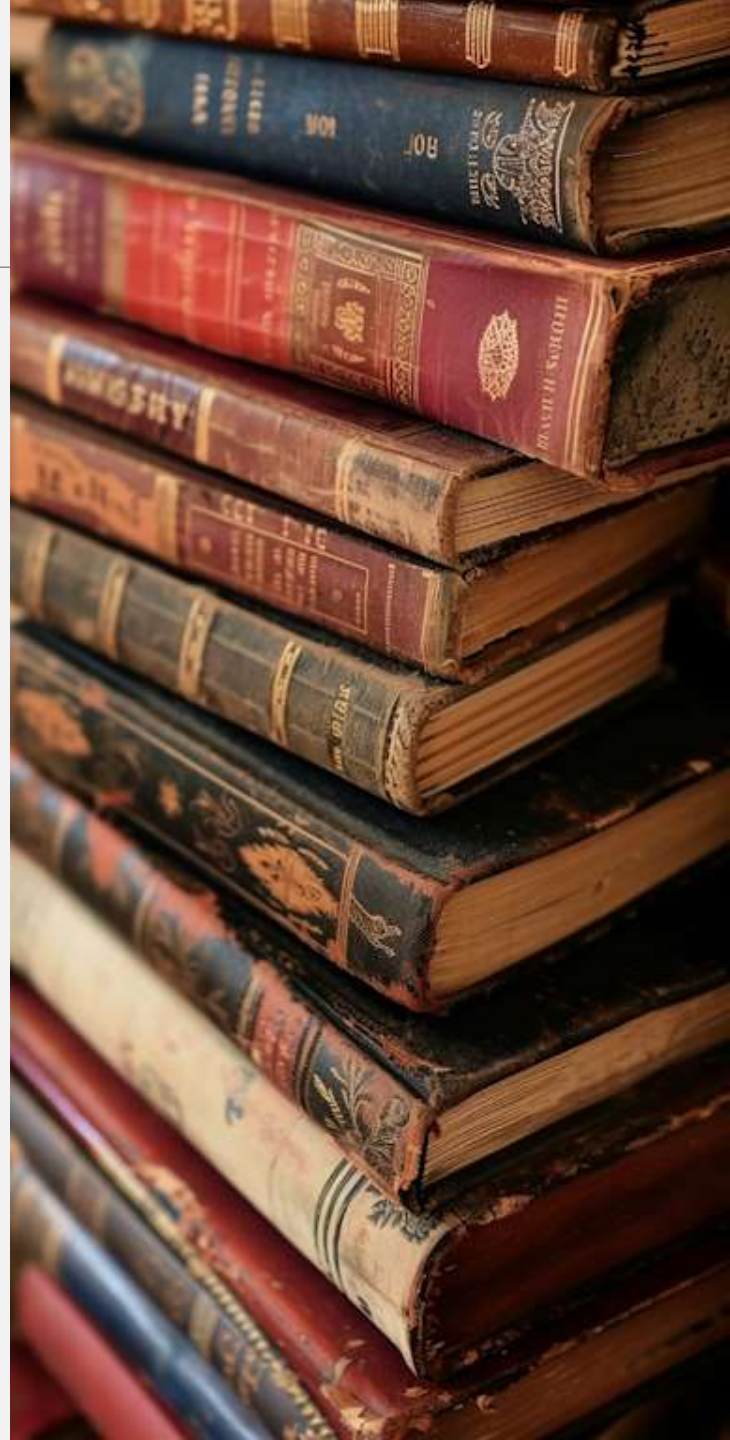
HEAD

The HEAD method asks for a response identical to that of a GET request, but without the response body. Used to check the availability of the resource.

Examples:

HEAD /users

HEAD /users/1



POST

The POST method sends a request for the server to accept the entity contained in the request as a new object identified by the sent URL.

Example:

POST /users

```
{
  "firstname" : "Jan",
  "lastname" : "Kowalski"
}
```

PUT

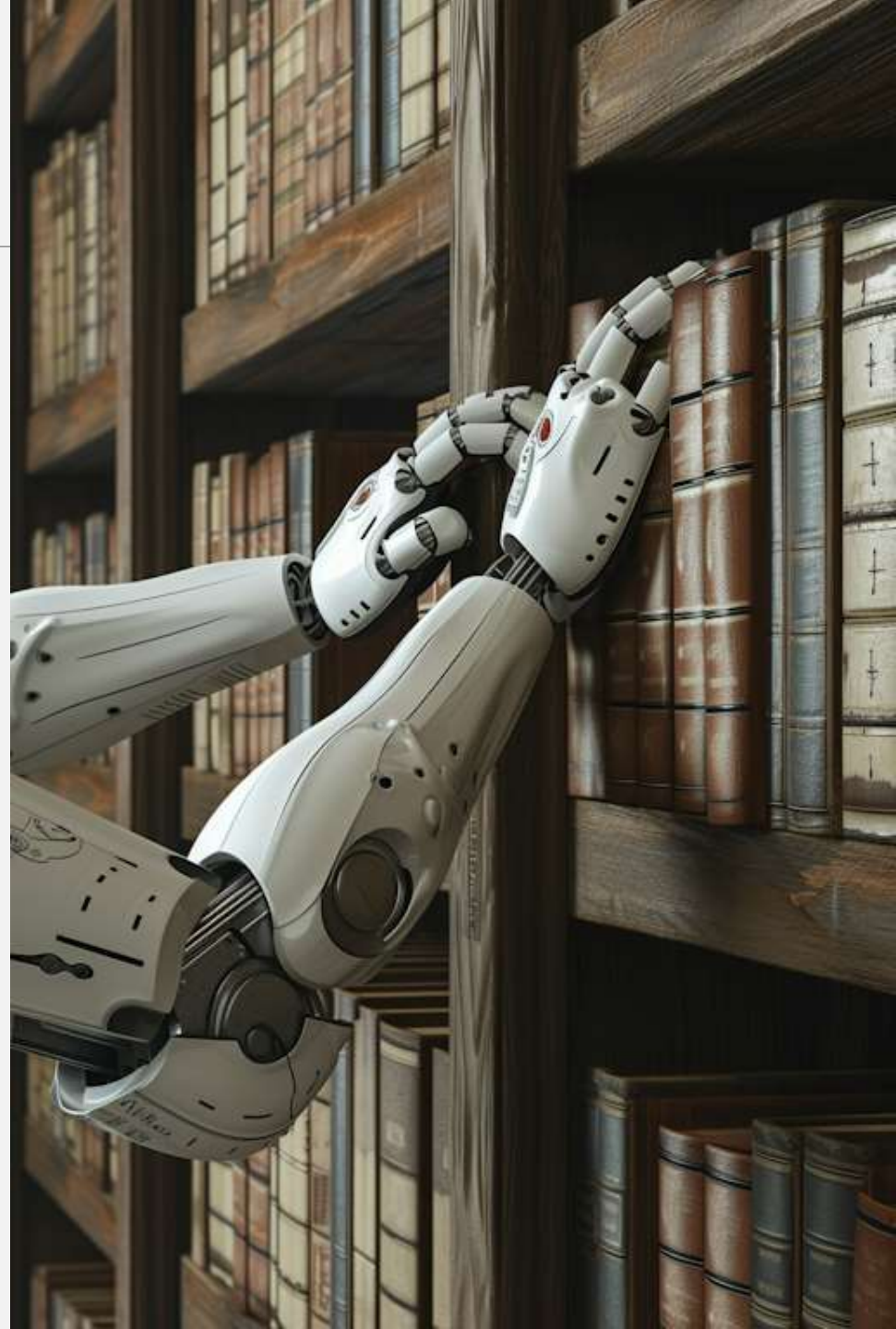
The PUT method sends a request for the object to be stored under the provided URL identifier.

If the URL identifier refers to an already existing resource it is modified, if the URL identifier does not point to an existing resource then the server can create a resource with that URL identifier.

Example:

PUT /users/10

```
{
  "firstname" : "Jan",
  "lastname" : "Kowalski"
}
```



PATCH

The PATCH method is used

for partial modifications of the resource

Example:

PATCH /users/10

```
{
  "lastname" : "Nowak"
}
```

DELETE

The DELETE method deletes a specific resource

Example:

DELETE /users/1

OPTIONS

The OPTIONS method returns the HTTP methods supported by the server for the specified URL.

Examples:

OPTIONS /users

OPTIONS /users/1

TRACE

The TRACE method performs a message loop-back test along the path to the target resource, providing a useful debugging mechanism.

Examples:

TRACE /users

TRACE /users/1

CONNECT

The CONNECT method establishes a tunnel to the server identified by the target resource.

Idempotency

Multiple method calls under the same initial conditions will always have the same result.

METHOD	IDEMPOTENT
GET	YES
HEAD	YES
POST	NO
PUT	YES
DELETE	YES
CONNECT	NO
OPTIONS	YES
TRACE	YES
PATCH	NO *

Safety [not security]

No data modification or side effects.

METHOD	IS SAFE
GET	YES
HEAD	YES
POST	NO
PUT	NO
DELETE	NO
CONNECT	NO
OPTIONS	YES
TRACE	YES
PATCH	NO

Cacheability

A cacheable response is an HTTP response that can be cached, that is stored to be retrieved and used later, saving a new request to the server.

METHOD	IS CACHEABLE
GET	YES
HEAD	YES
POST	YES *
PUT	NO
DELETE	NO
CONNECT	NO
OPTIONS	NO
TRACE	NO
PATCH	NO

Body

Not every method can use body

METHOD	REQUEST HAS BODY	RESPONSE HAS BODY
GET	OPTIONAL *	YES
HEAD	OPTIONAL	NO
POST	YES	YES
PUT	YES	YES
DELETE	OPTIONAL	YES
CONNECT	OPTIONAL	YES
OPTIONS	OPTIONAL	YES
TRACE	NO	YES
PATCH	YES	YES

GET case: https://evertpot.com/get-request-bodies/

HTTP headers

- contain meta-data for the request
- may be present in any method
- have form key:value
- there are headers with defined meanings that should result in specific behavior of the receiving side (client or server)
- developers can define their own headers and give them meanings
- ~~custom ones should contain x- prefix~~
- there are no restrictions in the standard, but most servers only allow a certain number of headers of a predefined size

* letter size of the headers is ignored.

HTTP headers: request examples

- Accept – specifies the client's acceptable response format. Accept: application/json
- Accept-Language – customer-accepted response language. Accept-Language: pl-PL
- Host – server domain hame. Host:wp.pl
- User-Agent – client identifier. Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.101 Safari/537.36
- Cookie – cookie related to request

HTTP headers: response examples

- Server – the name of the server. Server:Apache/2.4.10 (Unix) OpenSSL/1.0.1e-fips mod_bwlimited/1.4
- Date – date and time when response was sent. Date:Sat, 21 Oct 2017 10:44:53 GMT
- Content-type – type of returned data. Content-type: text/html; charset=UTF-8
- Set-cookie – command for the browser to set a value in a cookie.
- Content-length – response size in bytes
- Location – address server wants the browser to redirect to Location: https://www.wp.pl/

HTTP content type

Content Type (MIME) – Multipurpose Internet Mail Extensions

- application/json
- application/xml
- image/jpg
- text/plain
- … and many more

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types

HTTP status code

- 1XX - information codes
- 2XX - success codes
- 3XX - redirect codes
- 4XX - client application error codes
- 5XX - HTTP server error codes

1xx Informational

- 100 Continue
- 101 Switching Protocols
- 102 Processing
- 103 Early Hints
- 110 Connection Timed Out
- 111 Connection refused

2xx Success

- 200 OK
- 201 Created
- 202 Accepted
- 204 No Content

3xx Redirection

- 301 Moved Permanently
- 302 Moved Temporarily
- 303 See Other
- 304 Not Modified

4xx Client error

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 406 Not Acceptable
- 408 Request timeout
- 409 Conflict
- 413 Payload too large
- 415 Unsupported Media Type

5xx Server error

- 500 Internal Server Error
- 501 Not implemented
- 502 Bad Gateway
- 503 Service Unavailable
- 504 Gateway Timeout
- 505 HTTP version not supported

HTTP status code ☺

- 402 Payment required
- 418 I’m a teapot
- 451 Unavailable For Legal Reason

HTTPS

https://sekurak.pl/czy-zielona-klodka-w-przegladarce-https-certyfikat-ssl-chroni-przed-czymkolwiek/

https://howhttps.works

Default protocol https is used by 85.2%

of all the websites.

HTTPS

S in HTTP stands for „secure” ☺

HTTP is not a secure protocol - it sends information in plain text in an unencrypted way

This is sufficient to display simple pages or even complex services as long as they do not display or send sensitive data to them

The solution is to use the HTTPS extension (port 443)

HTTPS uses cryptographic mechanisms to ensure the confidentiality of transmitted information, both symmetric and asymmetric algorithms, as well as certificates and PKI

{your-favourite-browser-name} Dev Tools

It is used to preview requests, the DOM tree, JS code, security settings or performance testing

Allows you to simulate a poor connection, as well as the operation of the application completely offline

You can check how the application will behave on a particular screen resolution or on a cell phone

Usually can be turned on using

- F12
- Ctrl + Shift + I
- Right mouse buton click, menu item → inspect element

POSTMAN

Convenient yet powerful graphical tool for testing http communication

HTTP client as a desktop application.

Allows you to easily build HTTP requests using header methods, bodies.

Saves executed requests so it is easy to replay them.

https://www.postman.com/downloads/

Other tools

Fiddler - A tool that allows you to monitor, intercept, debug and modify HTTP/HTTPS traffic between your computer and web applications.

It is able to make the life of network administrators much easier, all connections are described in detail and the interface gives you the possibility to set up breakpoints or use scripts, among other things.

https://www.telerik.com/fiddler#fiddler-classic

Wireshark – free and open-source packet analyzer. It is used for network troubleshooting, analysis **[do not install it on company computer without permission]**

https://www.wireshark.org/download.html

Library

- https://developer.mozilla.org/en-US/docs/Web/HTTP
- https://tools.ietf.org/html/rfc2616

- Michał Zalewski – „Splątana sieć” / „The Tangled Web”

Sample APIs:

- https://github.com/public-apis/public-apis
- https://public-apis.xyz
- https://dane.gov.pl/pl/dataset