Sentinel

Networking - Protocols

DEFENDING OUR DIGITAL WAY OF LIFE

Today we will:

- Learn what is a network protocol
- Learn how two computers form a connection
- Get to know in depth an important protocol - HTTP

Client and server

- 2 roles in communication:
  - Client - requesting something
  - Server - answering

Hey, send me that cat photo

You got it

Client and server

Let's say the client (me) wants to start a conversation with Google server.

What's the first thing I need to know?

Client

Server

180.12.30.1

Client and server

Now that I know Google's IP, I need to know how to talk with the server.

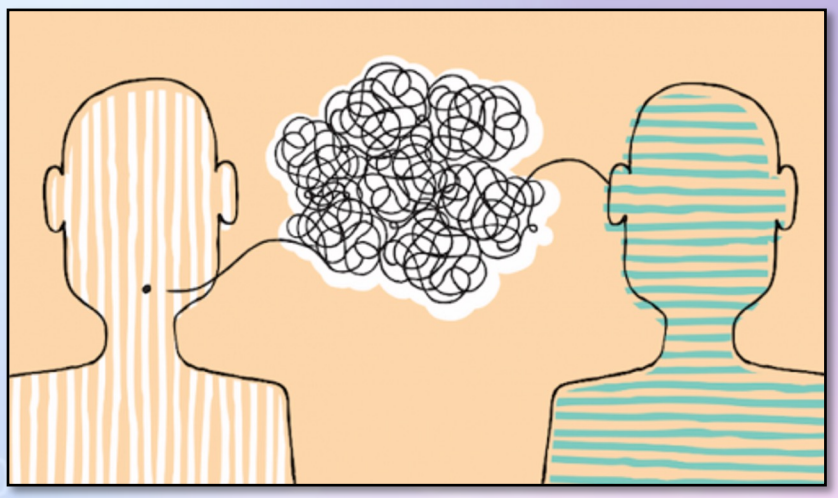
Client

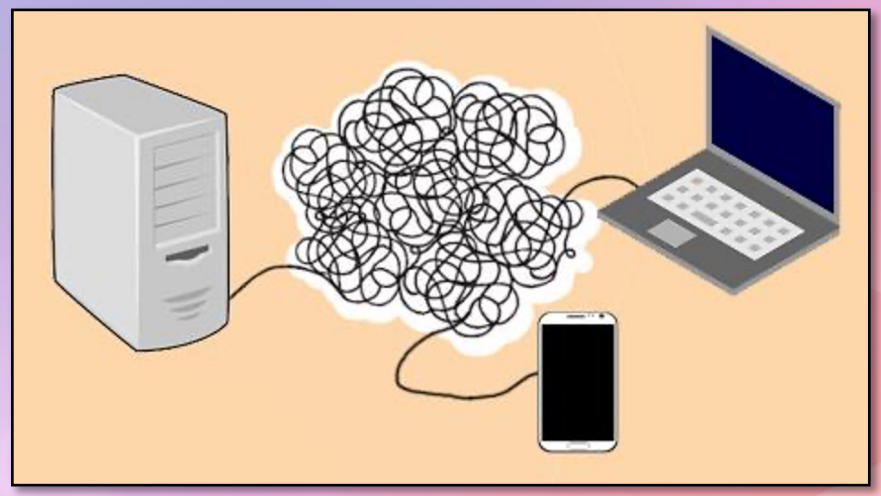
Server

180.12.30.1

Protocol

Protocol is a set of rules that allows devices to communicate with one another. Simply put, it's a language for computers.





Let's take a look at some human protocols…

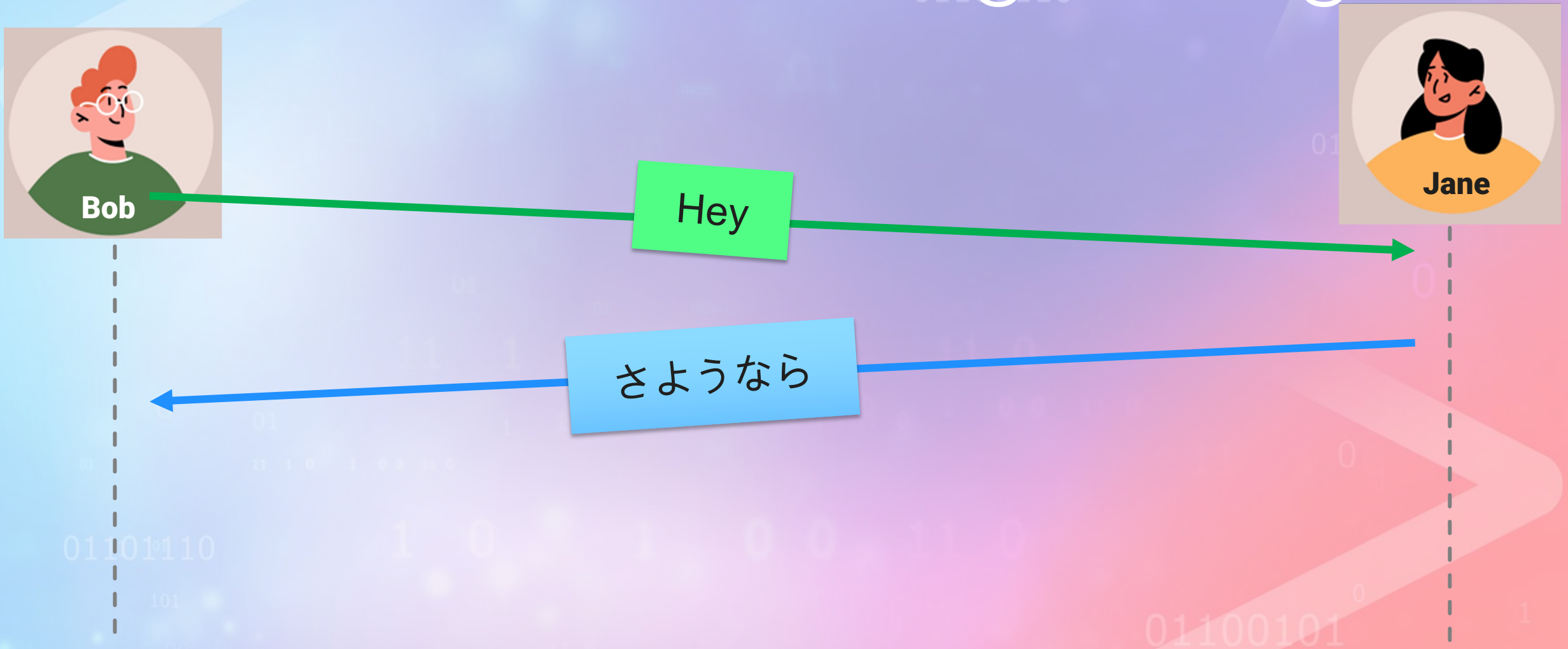
Hey

Hey

What's the time?

14:30

Can a conversation go wrong?



Computer protocols

- Like humans, computers speak many languages:

HTTP

DNS

SSH

FTP

Torrent

SMTP

- Don't worry! You'll be familiar with at least some of these very soon.
- Unlike humans, computer protocols have very strict rules.

Communication between computers

- So our client wants to start a conversation with our server.
- He needs to know:
- 1. The IP of google (we'll call it: **destination ip** or **dst ip**)
- 2. The protocol he wants to use.

Client

Dst IP: 180.12.30.1

HTTP

HTTP

Server

Communication between computers

But what if our server knows not one, but many protocols?

Client

Dst IP: 180.12.30.1

HTTP

HTTPS

SSH

HTTP

Server

Ports

The server assigns a number to each of the services he provides.

This is called a **port**.

Client

Dst IP: 180.12.30.1

HTTP

HTTPS

HTTP

SSH

Server

Ports

Now, our client has to choose the specific port he wants to communicate with.

Client

Dst IP: 180.12.30.1

Dst PORT: 80

HTTP

Port 443 HTTPS

Port 80 HTTP

Port 22 SSH

Server

Ports

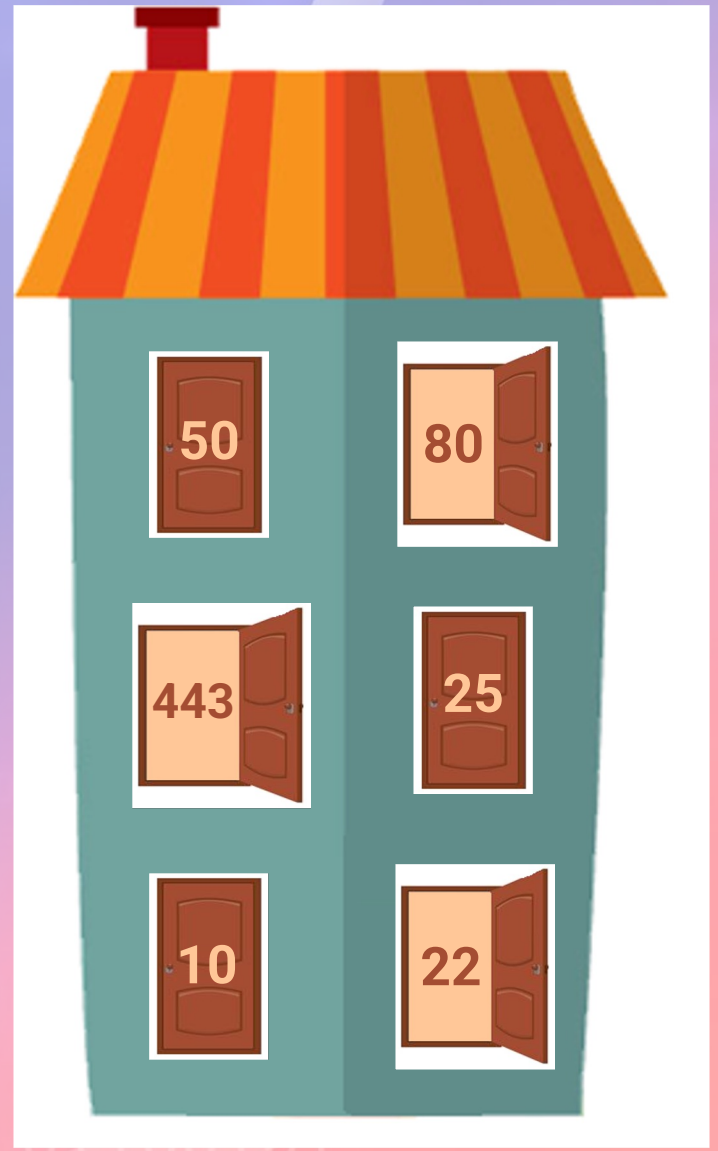
We can imagine each computer as a house, and each port as a door. Some ports are open (ready for connection) but some are close.

Client

Dst IP: 180.12.30.1

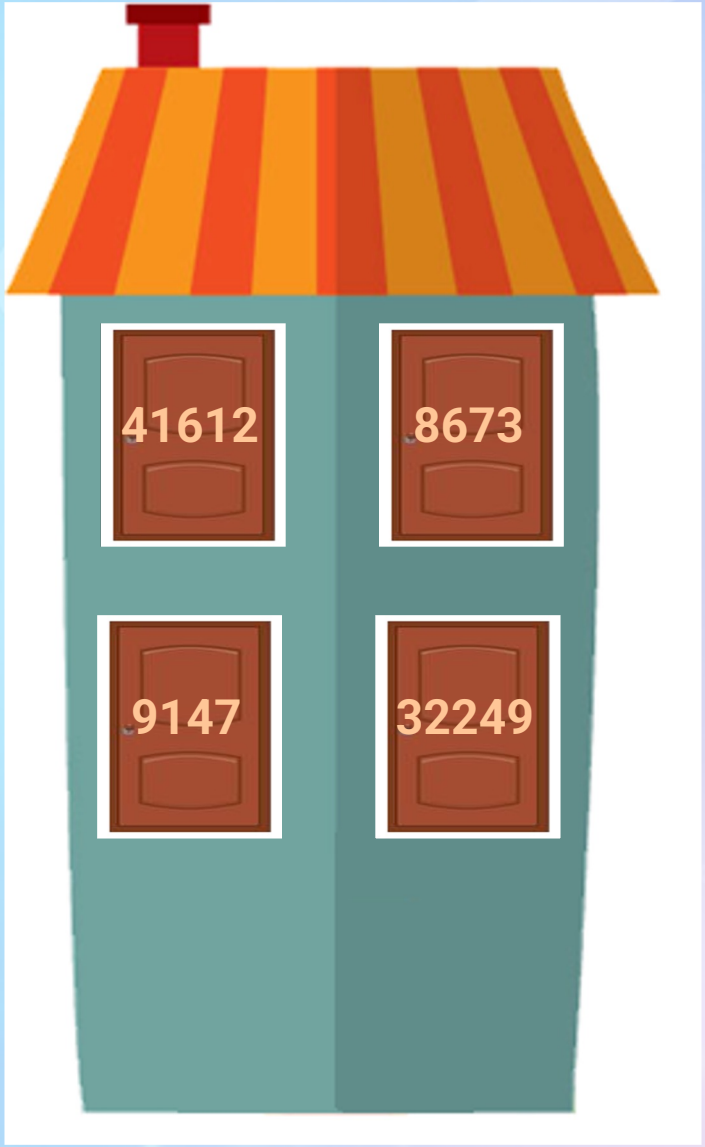
Dst PORT: 80

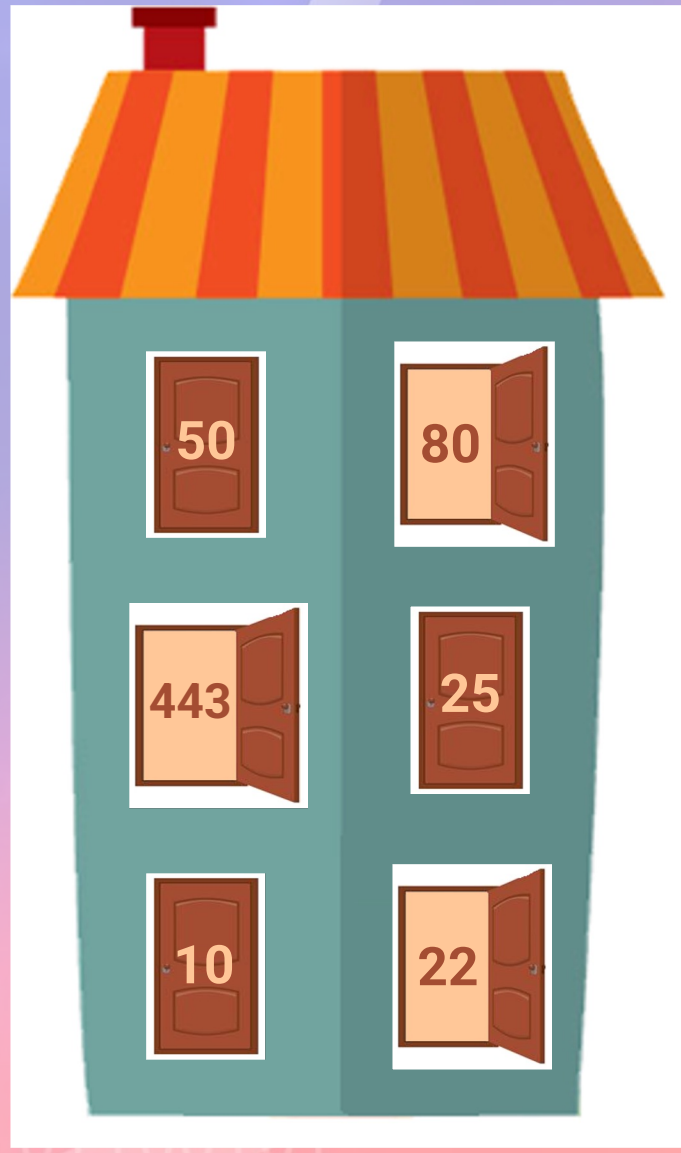
HTTP



Ports

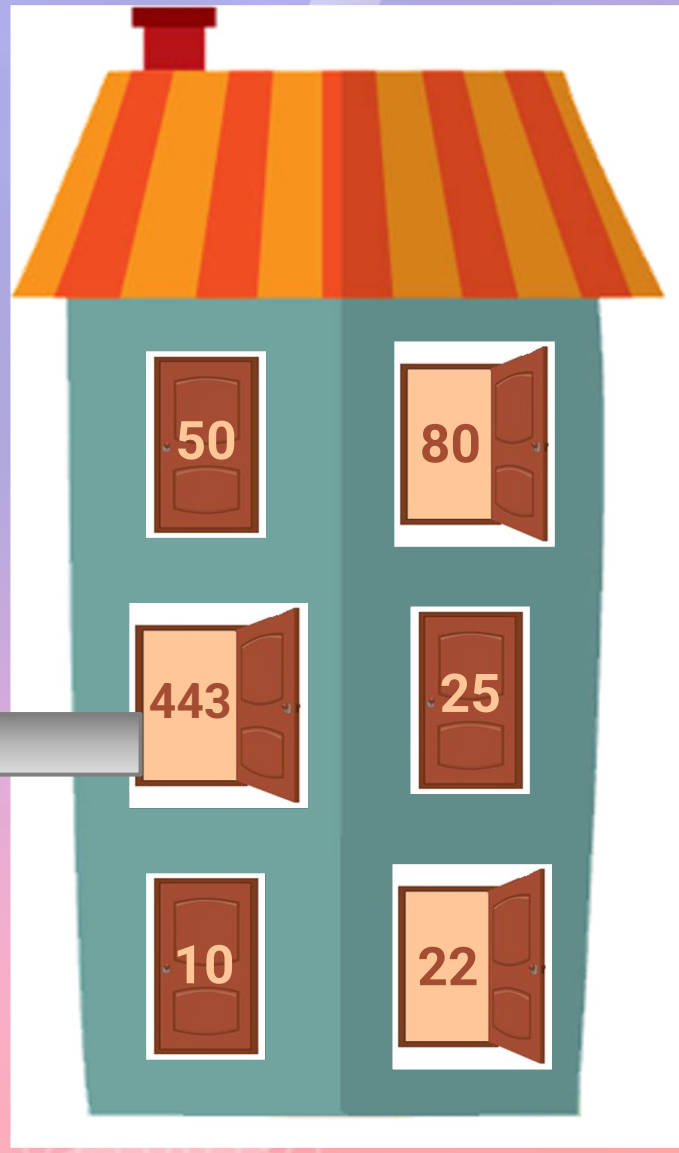
The client also has ports - those are the doors from which the messages are going out and in.





Ports

To start a new connection, the client opens one of his ports, and connects to an open port of the server.



Ports - Summary

Every connection has 4 main parameters:

- Source IP
- Destination IP
- Source Port
- Destination Port

IP - identifies the device

Port - identifies the specific application/connection on the device

HTTP

A protocol that allows transferring hyper-text (html, images, css, etc.)

Web sites and web applications (Google, Facebook) use HTTP

What can you tell about HTTP?

Usually uses the port 80 at the server side.

It's a **textual** protocol

- Readable text, not binary data.

HTTP Requests

- The most common request is GET.

HTTP Requests

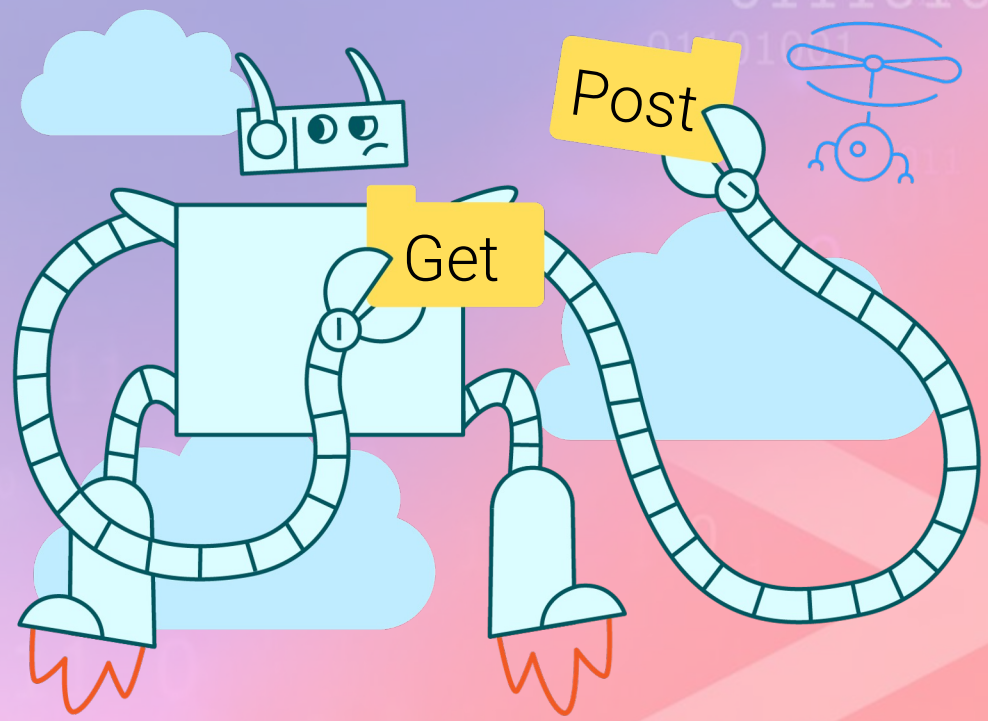
- When the client wants to send more data to the server, It can use POST Request.

```
POST / HTTP/1.1
Host: www.gmail.com
Content-Length: 27

username=user&password=pass
```

Requests types summary

- **GET** - "Hey server, please get this for me!"
- **POST** – "Hey server, please take this information and do something with it!"



HTTP Responses

An HTTP response contains the:

Status

e.g. success

Resource contents

e.g. the website HTML

HTTP Responses

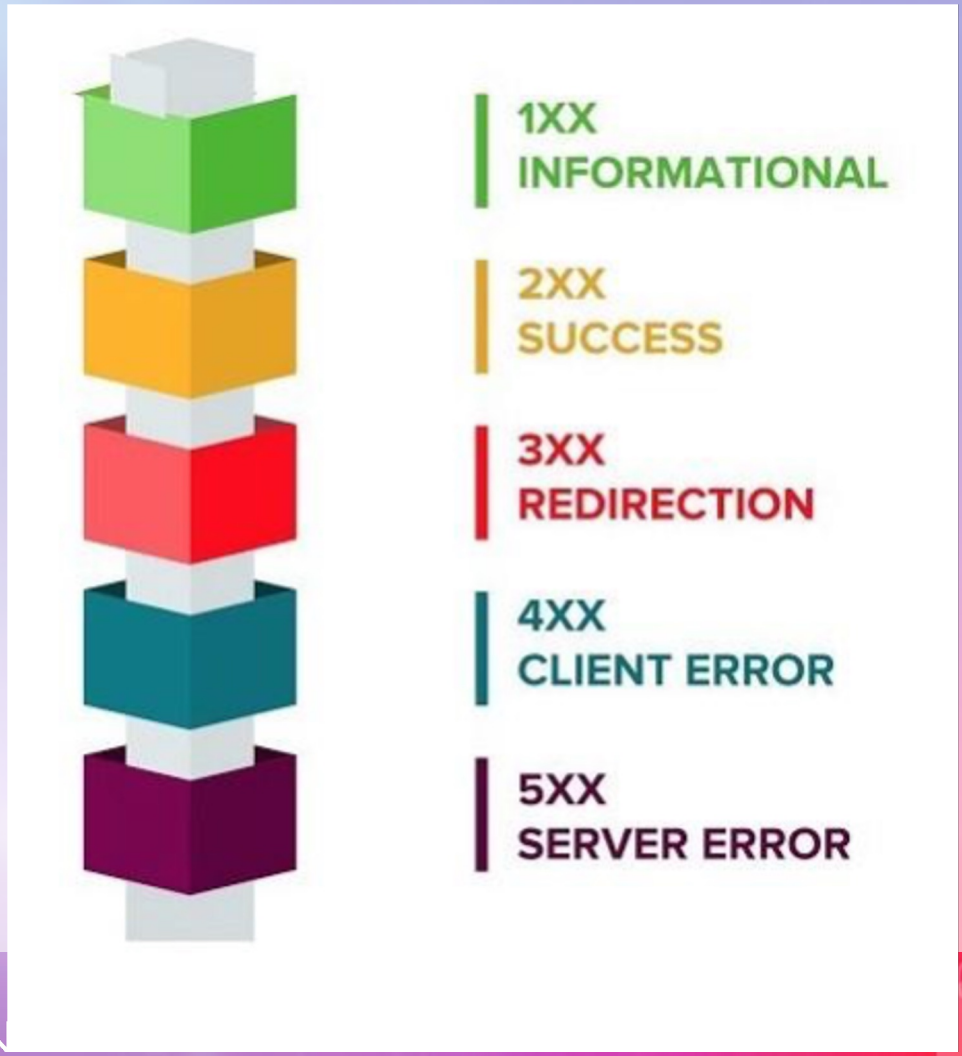
```
HTTP/1.1 200 OK
Content-

<html>
…
</html>
```

```
HTTP/1.1 404 Not Found
Content-Type: text/html

<html>
File not found!!!
</html>
```

HTTP Response status codes



HTTP Headers

- Key-value pairs sent in both HTTP requests and responses.

Host: www.google.com
key value

- Provide additional context for requests and responses.

Common HTTP Headers

Header	Purpose	Example
Host	Specifies the domain name of the server	Host: www.example.com
User-Agent	Identifies the client software making the request	User-Agent: Mozilla/5.0
Accept	Specifies the media types that the client is able to receive	Accept: text/html
Content-Type	Indicates the media type of the response	Content-Type: image/png
Authorization	Contains credentials for authenticating the client with the server	Authorization: Basic YWxhZGRpbjpvcGVuc2VzYW1l

User agent

- Sent by a web browser or other client application to identify itself to the server.
- Provides information about the client's software, operating system, and device, allowing the server to deliver an optimized response.

- Format:

  Software/Version (Operating System) Engine/Version

User agent - example

Mobile client

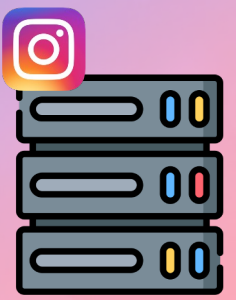
GET / HTTP/1.1
User-Agent: Mozilla/5.0 (Linux; **Android 10; Pixel 3 XL**)

200 OK
(Mobile version's HTML)

Desktop client

GET / HTTP/1.1
User-Agent: Mozilla/5.0 (**Windows NT 10.0; Win64; x64**)

200 OK
(Desktop version's HTML)

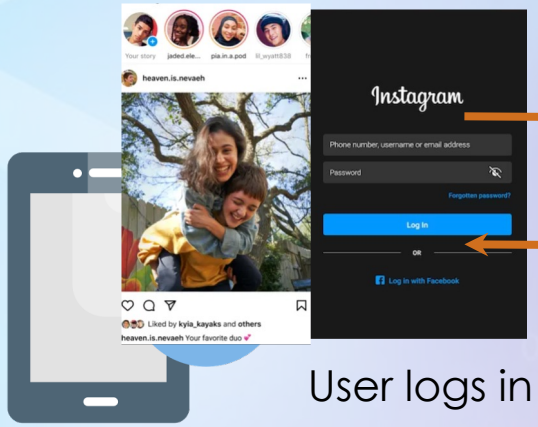


Server

Cookies

- HTTP is stateless - information is not kept between request
- What if we want to keep a state?
  - Keep a user logged in
  - Save user preferences (language, color theme, etc.)
- Use cookies!
  - Small text files stored on a user's device by a web browser.
  - Created by websites to store information.
- Cookies have expiration dates

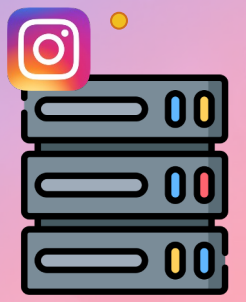
Cookies - example



User logs in

Client

I know this cookie, this user is already logged in



Server

GET / HTTP/1.1

200 OK
Set-Cookie: session_id=abc123;
Expires=Wed, 09 Jun 2023 10:18:14 GMT

Next time user uses the app:

GET / HTTP/1.1
Cookie: session_id=abc123

200 OK

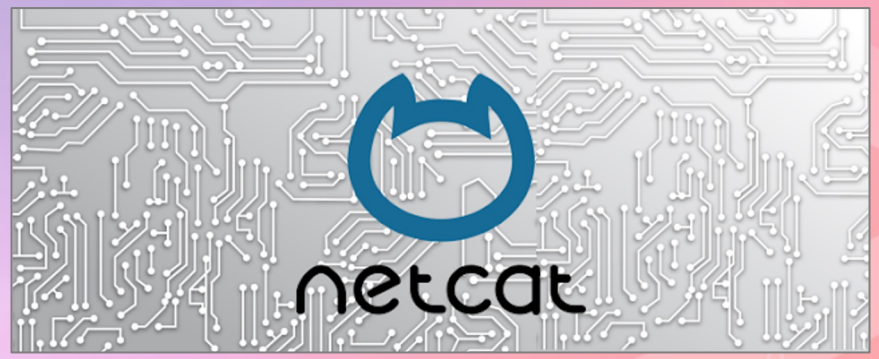
Talking HTTP

Because HTTP is a textual protocol, human beings can also speak it!

To do that, we will use a program called Netcat.

Netcat

- Command line networking tool.
- "The Swiss Army Knife of Networking".
- We will mostly use netcat to create connections to servers, and just… talk with them!



Netcat

- To open a new connection to a server with specific dst port:

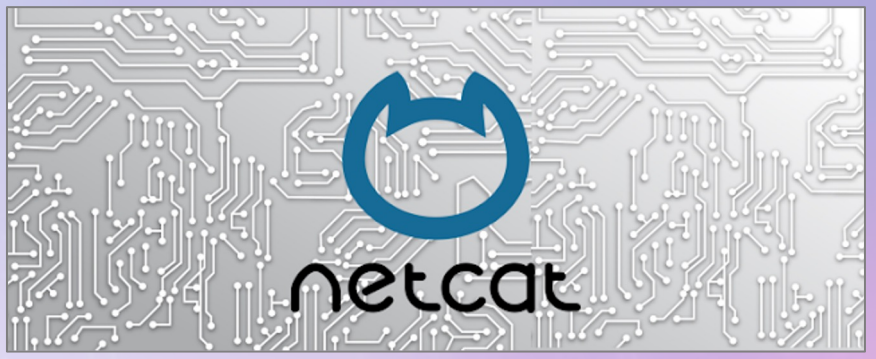
```
C:\> nc -v <IP\HOST> <PORT>
```

- Example:

```
C:\> nc -v google.com 80
```

Netcat demo

- Let's try to send an HTTP request to google.com together.



Talking HTTP with Python

We can also use Python to talk HTTP!

Requests

- Python library for HTTP requests.
- Let's examine it line by line:

```
import requests

url = 'http://example.com'
response = requests.get(url)

if response.status_code == 200:
    print(response.text)
else:
    print(f'Request failed: {response.status_code}')
```

What did we learn?

- Client/server
- Protocols
- Ports
- HTTP
  - Request
  - Reply
- Connecting with: Netcat, Python (requests library)