Web Mapping & Visualisation

The Web’s Architecture and Economy

Elisabetta Pietrostefani

Today

- A (brief an opinionated) history of the Web
- The server/client model
- The modern web mapping eco-system

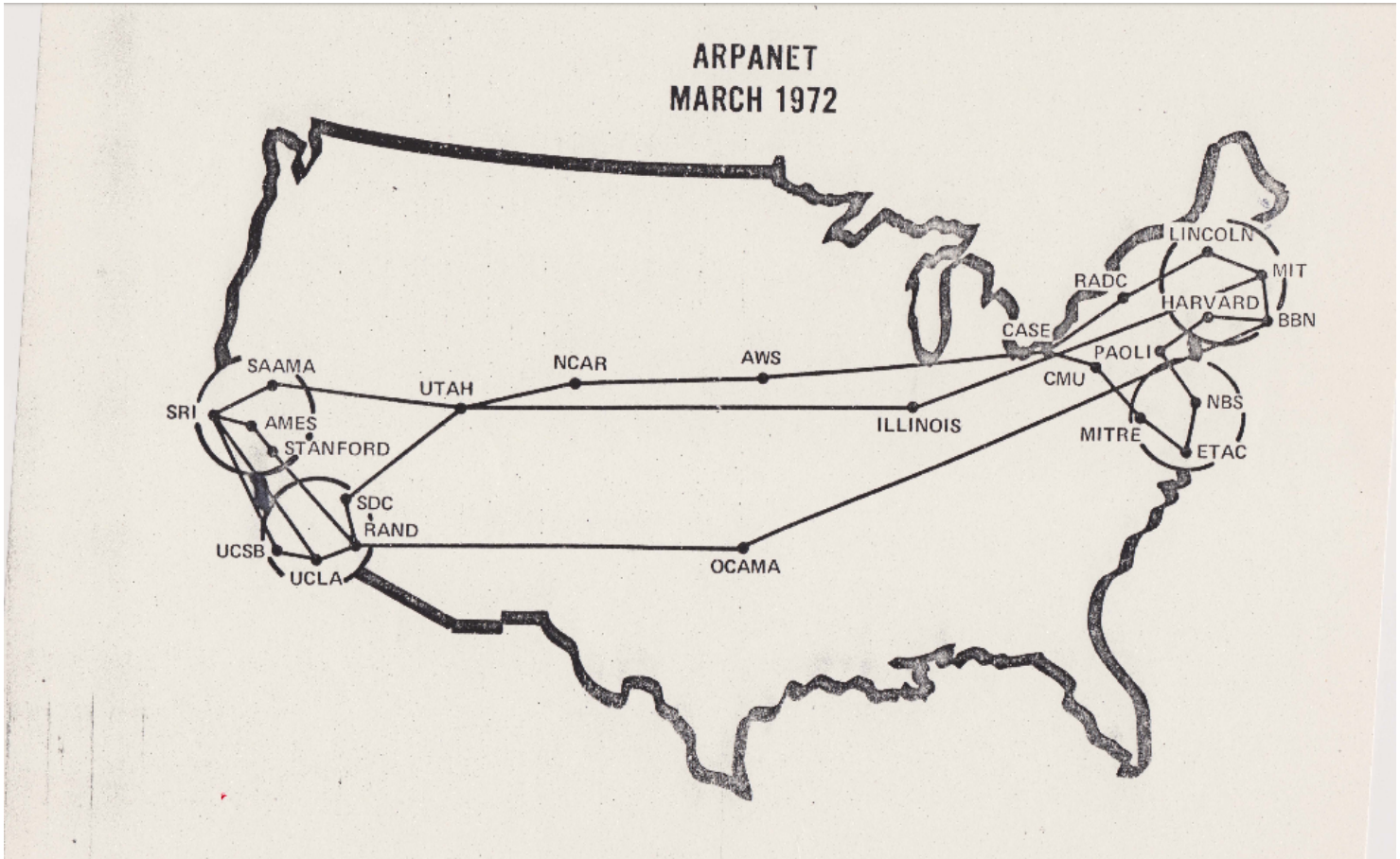
A (brief and opinionated) history of Web trends

Pre 1970s

The seeds:

- US (e.g. Licklider's "Galactic Network")
- Mostly military contracts (e.g. D/ARPA –> ARPANET) and "research'y"
- Develop protocols for machine communication

1970s - Birth of the *internet*



1980s

- Growth of the “web”
- From experimental validation to scaled up insfrastructure
- Free software (e.g. “Free as in Freedom”)

1990s

- Civilian and commercial growth
- Web 1.0
- Open Source software (e.g. “The cathedral and the Bazaar”)

2000s

- Web 2.0
- Mobile
- Web mapping takes off (hello Google Maps!)

2010s

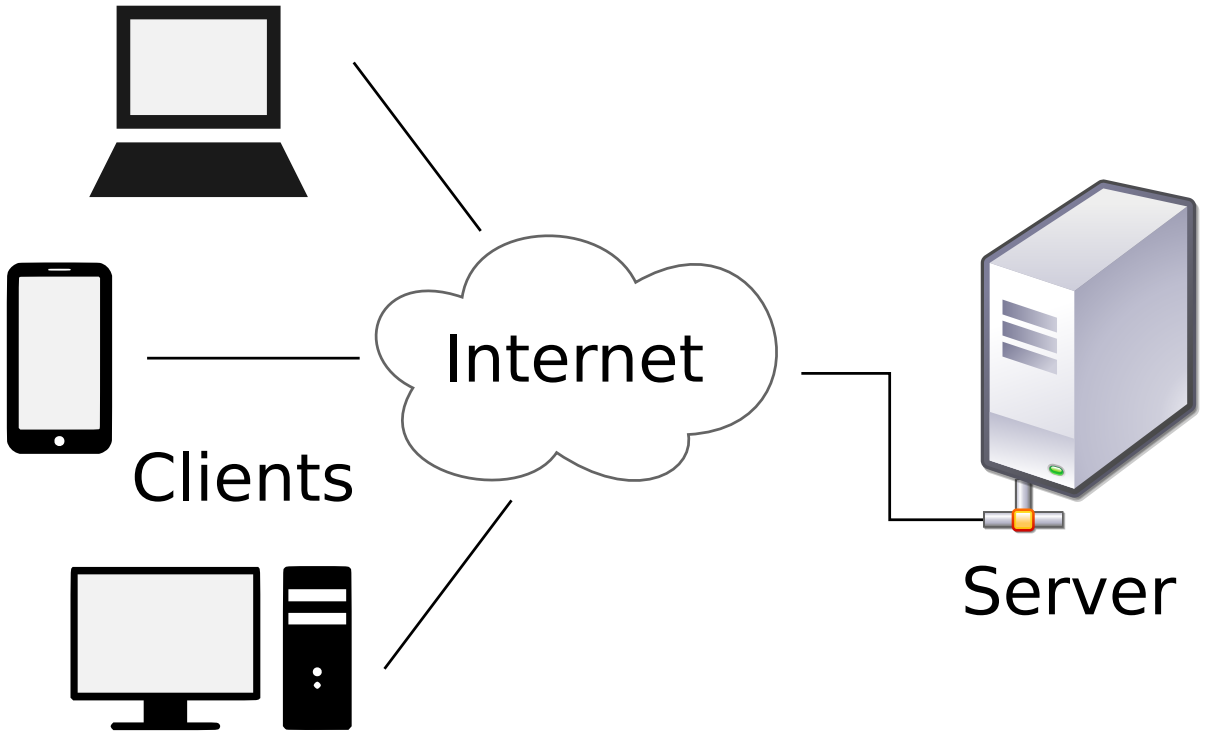
- Consolidation of GAFA –> concentration
- IoT
- Death of the desktop?

Ideas to retain

- The Web is technology to build decentralised systems
- Economics (for the most part) have turned it into a concentrated economy
- Computing today is physically distributed but socio-economically concentrated

The server/client model

The server/client model



Benefits

- Interoperability of disparate platforms
- Optimise on hard/software for each task (“distribute”)
- Separate data collection (e.g. sensor), storage (e.g. data centre), intensive computing (e.g. compute cluster), interaction (e.g. mobile)

Disadvantages

- Requires (cheap & ubiquitous) connectivity
- More complex than an isolated approach (e.g. desktop)
- Harder to “keep afloat”

Building blocks of a web map

Backend	*Frontend*
	
Server	Client
Data, mapping (GIS)	Style (CSS), web (HTML)

The current web mapping landscape

- **Software**: a lot of open-source projects
- **Platforms**: a concentrated few (web infrastructure is hard and expensive!)
- **Business model**: software as a service

The current web mapping landscape

Trade-off between

convenience + agility

Vs

flexibility + ownership

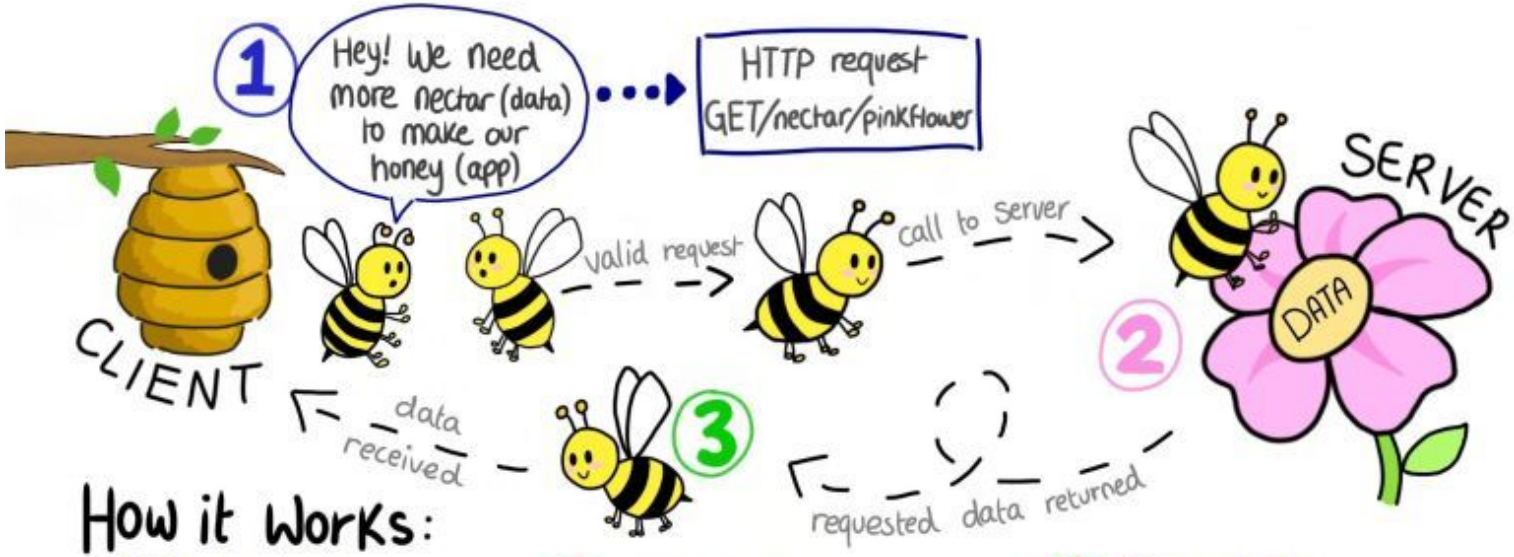
This course: mostly rely on commercial platforms to focus on design and cartographic rather than engineering concepts

What do APIs actually do?

What is an API?

@Rapid_API

An application programming interface allows two programs to communicate. On the web, APIs sit between an application and a web server, and facilitate the transfer of data.



How it Works:

1 Request
API call is initiated by by the Client application via a HTTP request

2 Receive
Our worker bee acts as an API, going to a flower (server) to collect nectar (data)

3 Response
The API transfers the requested data back to the requesting application, usually in JSON format

What do APIs actually do?

- Application Programming Interfaces (“APIs”)
- Instead of downloading a data set, APIs allow programmers, statisticians (or students) to request data directly from a server to a local machine.

When you work with web APIs, two different computers - **a client and server** - will interact with each other to request and provide data, respectively.

RESTful Web APIs are all around you.

Web APIs

- Allow you query a remote database over the internet
- Take on a variety of formats
- Adhere to a particular style known as Representation State Transfer or REST (in most cases)
- RESTful APIs are convenient because we use them to query database using URLs

Two ways to collect data

Plug-n-play packages. Many common APIs are available through user-written Py libraries.

Writing our own API request. If no wrapper function is available, we have to write our own API request and format the response ourselves.