Dissecting the Web

Deep Dive into the web mechanics

Mario Hunter - 13 October 2021

A web Demo

Getting a dog picture on your mobile

A Historical View

When the Web was born? 1989

Sir Tim Berners-Lee, a British scientist, invented the World Wide Web (WWW) in 1989, while working at CERN. The Web was originally conceived and developed to meet the demand for automated information-sharing between scientists in universities and institutes around the world.
The basic idea of the WWW was to merge the evolving technologies of computers, data networks and hypertext into a powerful and easy to use global information system.

How the web started?
Tim Berners-Lee wrote the first proposal for the World Wide Web in March 1989 and his second proposal in May 1990. Together with Belgian systems engineer Robert Cailliau, this was formalized as a management proposal in November 1990. This outlined the principal concepts and it defined important terms behind the Web.

By the end of 1990, Tim Berners-Lee had the first Web server and browser up and running at CERN.

Check out the first website ever made!
http://info.cern.ch/hypertext/WWW/TheProject.html

Open standards
An essential point was that the web should remain an open standard for all to use and that no-one should lock it up into a proprietary system. In this spirit, CERN submitted a proposal to the Commission of the European Union under the ESPRIT programme: “WebCore”. The goal of the project was to form an international consortium, in collaboration with the US Massachusetts Institute of Technology (MIT). In 1994, Berners-Lee left CERN to join MIT and founded the International World Wide Web Consortium (W3C). Meanwhile, with approval of the LHC project clearly in sight, CERN decided that further web development was an activity beyond the laboratory’s primary mission. A new European partner for W3C was needed.

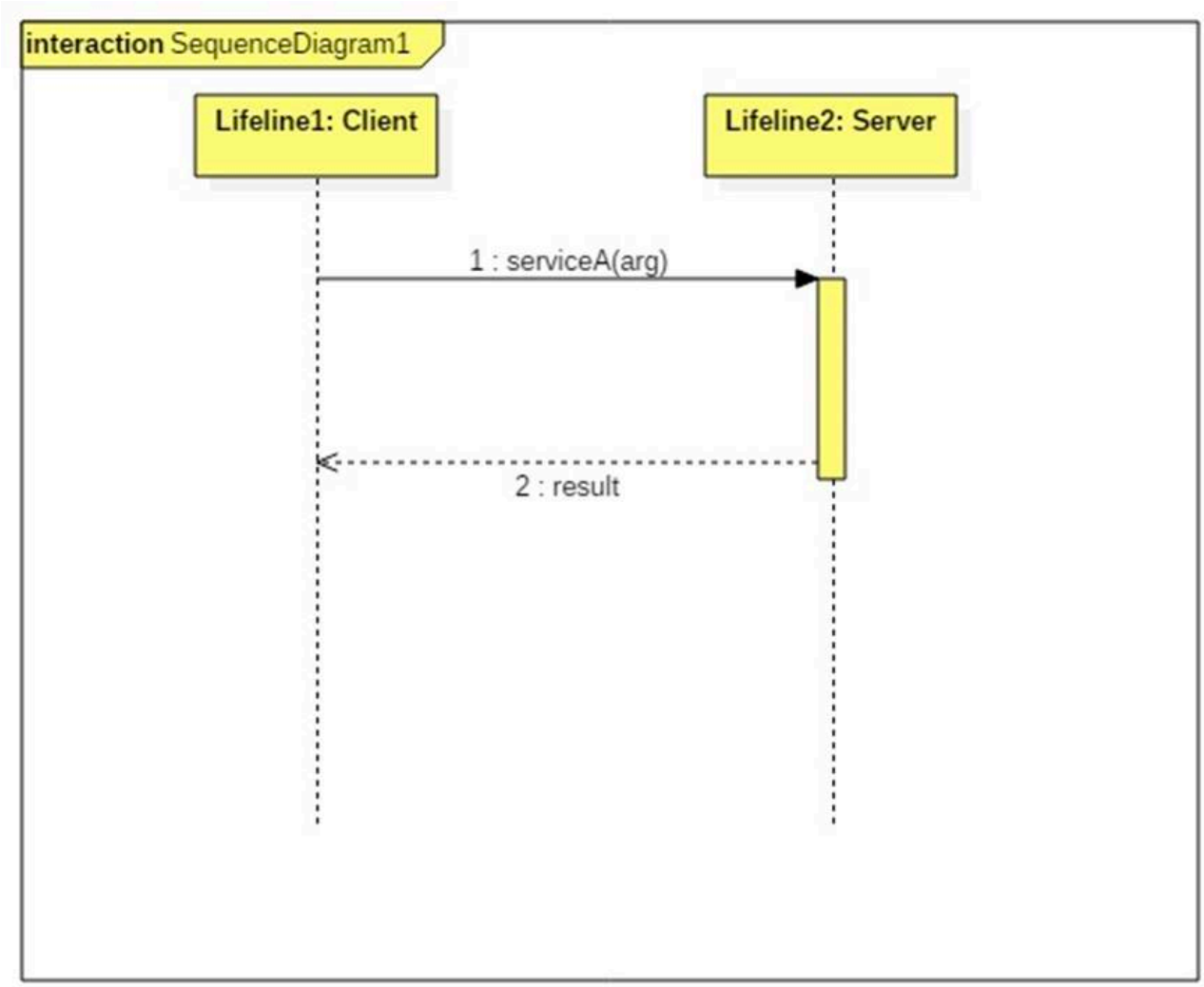
A System View

How the web works?
Content is hosted on a *machine* somewhere. Then, what happens is that *someone* wants to access this content. So, the *someone* makes a *request* to this *machine* stating who he is, and what content he wants. After that, the *machine* responds to the *requests* with all the data the client wished for.

That is simply put as a Client-Server model.

The client is the *someone* asking for requests.

The server is the machine - since it serves the client's requests.



Client-Server Model Sequence
1. Client issues a request to the server
2. Server *processes* the request
3. Server responds to the client with a response

The client

A client is any application making requests on behalf of the user. It initiates requests receives responses, render them. An important distinction to make the client in the web model is not the user himself. The client is the app itself issuing the requests. For example, web browsers are clients, a http library for mobile apps is a client, a linux package making requests is a client. In some cases, the term *Client* is used to denote to the machine running the software making web requests

The server

A server is likewise a piece of software that listens for clients' requests waiting around to serve them by responding with the corresponding content. Examples for web servers include TomCat, Xamp, Lamp, Wamp, Microsoft Server. And again, the term server can also be used to denote the machine running the server software.

The connection

The client and server communicates over the network using a protocol designed by the aforementioned Tim Berners-Lee called HTTP. The protocol describes in detail how the client and server should communicate - more on that later.

The mixup

Client and Server are merely roles in the process. A single application can be a client to some server, and a server to another client.

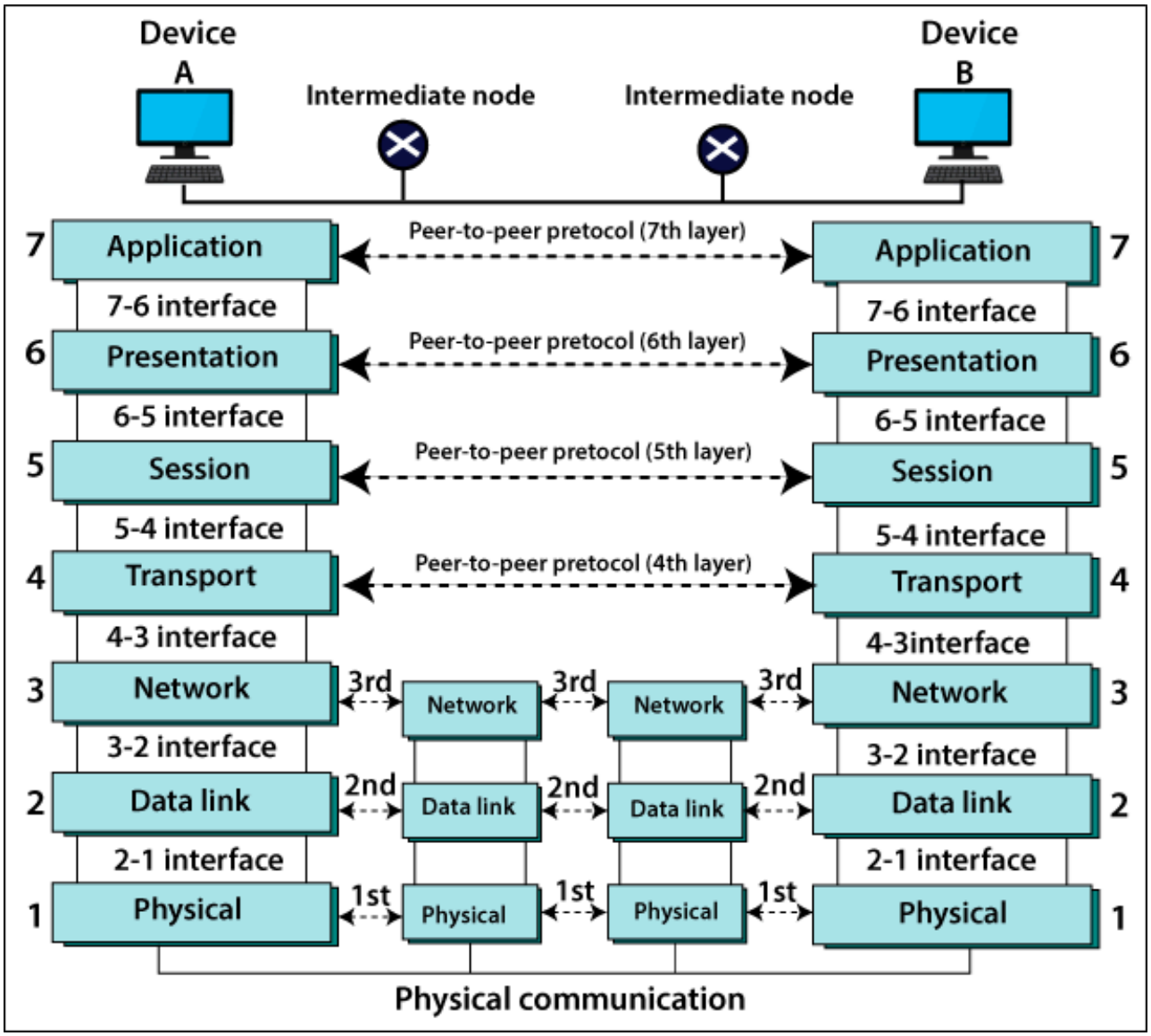
To make matters more concrete, imagine a website letting the user signup. Included in the registration information is the user's country. So here, the client - the web browser, sends the information to the server which saves the data. Then, when the user logs in the server responds with the home page of the user. In the homepage is all the information the

user entered during registration. In addition, it has the current temperature of the user's country. So how our website server got ahold of this info?

Simply put, it made a *request* to another public server asking for the current temperature of this country. In other words, our *server* became a *client* to the *weather server*.

A network View

To communicate together the client makes a *text* request that is being passed around a 7-layer stack to reach the server where it will be passed up the 7-layer again.



We are interested only in the first layer - the application layer.
To communicate the application makes what is known as a HTTP request.

A Network View L2

Earliest version was HTTP v0.9
A simple HTTP Request is made to the server requesting content, and the response is the content written in HTML.

Telnet demo

- $> telnet google.com 80
- Connected to 74.125.xxx.xxx
-
- GET /
-
- (hypertext response)
- (connection closed)

A Historical View L2

1- Static hosting
Files were hosted and physical paths were exposed and files are served
Like the first website

2- server side content rendering
Files were computed at runtime and files are generated and returned to client.
Each action goes back to the server to be processed and the new page is returned to the client.

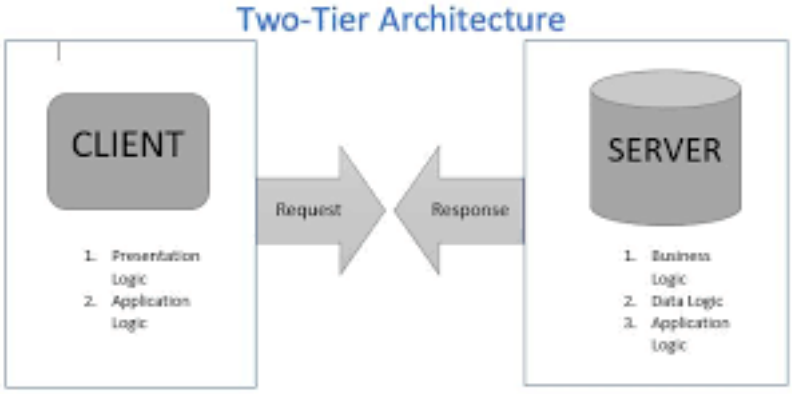
2- client side scripting
Some fancy logic can be added to the client side (styling for example), collapse a row
Even more actions can be done all client side with server coordination in the background

https://wordpress.com/themes

3- Recent progressions, SPA, PWA — more on that in the next section
Examples: Facebook, twitter ..

A System View L2

Two-Tier Architecture
A two-tier architecture is a software architecture in which a presentation layer or interface runs on a client, and a data layer or data structure gets stored on a server. Separating these two components into different locations represents a two-tier architecture, as opposed to a single-tier architecture.



This is the model we will use in the assignments.
We have a client app that contains all the presentation logic, and a server app which will contain business and data logic and maintains our system state.

API

3-tier

Demos