NETWORK PROTOCOL

msgs sent over wire or network ≡ msgs sent over the internet from 1 machine to another machine

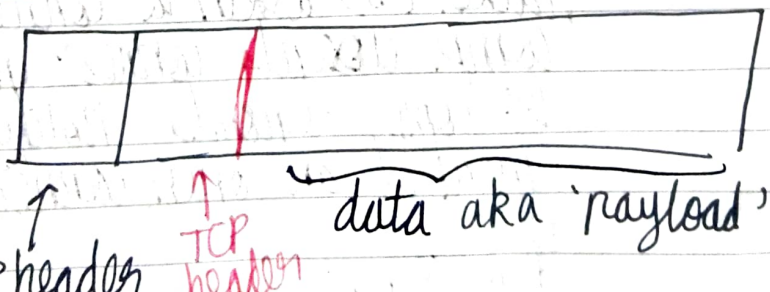
Network protocol consists of:
- types of msgs
- struc.
- order
- response to a msg, what shud it look like if present
- rules on when msgs can be sent to 1 another

3 main protocols:

1) IP → Internet Protocol

i) Modern internet effectively runs on IP. When 1 machine (client) interacts with another (server) and sends data to it the data is sent in form of IP packet (fundamental unit bldg blocks of data sent b/w machines i.e. communication)

ii) IP packet → made up of bytes



⇒ IP packet

iii) IP header - At beginning of packet. Contains imp info → source IP address, destination IP address (machine sending data) (machine data will go to)

destinaⁿ ≡ dstn

total packet size: version of IP that packet is operatin by (current used → IPv4, IPv6). Based on IP version, packet might looks diff
header size : 20 to 60 bytes

iv) Size of IP packet ≈ 2^{16} B ≈ 65 KB
∴ info sent as multi-IP packets across machines. Order in; assurance that packets will be sent is kinda fked.
That's why, TCP is req-d.

2) TCP – Transmission Control Protocol
Implemented in the kernel, which exposes sockets to apps, that they can use to stream data thru open connecⁿ
i) Built on top of IP. Solves order, assurance issues of IP packets in an error-free, uncorrupted way
ii) Used in web apps mostly
iii) TCP header → present in data porⁿ of packet
Contains TCP info like order of packets
eg:- browser wants to connect to a dstn server such as Google. Browser is first gonna create a TCP connecⁿ with dstn comp. / server thru a handshake
iv) Handshake : Special TCP interacⁿ where 1 comp communicates with other by sending packet(s) askin to connect. Receiving comp. accepts connecⁿ invite.
1st comp responds notifyin establishment of an open connecⁿ b/w 2 of em.
Now machines can exchange data.
v) If 1 of the machines doesn't send data in a given time period, connecⁿ can be timed out

devnr = developer

vi) A machine can end comm. by sendin a msg notifyin its intenⁿ and then TCP connecⁿ is done

vii) Thus, TCP is a more powerful, funcⁿal wrapper around IP.
It lacks robust framework that developers use to define comm. channels b/w client-server in a sys. cuz its just data that fits into the underlyin IP packets

3) HTTP - Hyper Text Transfer Protocol

i) Built on top of TCP
ii) Introduces higher lvl abstracⁿ (req-respons paradigm) above TCP
iii) 1 machine req. 1 machine responds. Makes it easy for devnrs to create easy to use, robust syss
iv) Req. : Machine that wants to interact with other machine sends this. Hv lotta params. defined by HTTP
v) Resp : Destn machine's reply to a reqy
vi) Req., Resp → can be thought to be similar to objs with imp fields, params. that describe em.

Can be visualized (not exactly like this in reality) as

```
const httpReq = {
    host: 'localhost',         // describe destn. server
    port: 8080,
    method: 'POST',  //GET, PUT, DELETE, OPTIONS, PATC
    path: '/payments',         // here used to POST payments i.e. pay
    headers: {
        'content-type': application/json,
        'content-length': 51
    }
    body: '{"data": "This is a low JSON format data piece"}'
}
```

POST → provides data → server
GET → retrieve data → server
DELETE → delete data

describe purpose of req. They're guidelines subj to your server and how we em.

path → Client sends data to server which has various paths which dictate what kinda logic will occur for req. eg: auth

```
const httpResponse = {
    statusCode: 200,
    headers:
        {
        'access-control-allow-origin': 'https://www.goog
        'content-type': 'application/json',
        },
    body: '{}'
}
```

describes type of resp. status codes are also like guidelines that you or alter as per reqmt
eg: - 404 status code = requested data piece not found

Path $\xrightarrow{\text{contains}}$ logic $\xrightarrow{\text{gets executed}}$ as per path provided in req

Headers → collection of k-v pairs containing imp. meta data or info abt req.

eg: 403 status code → data you're req. in is forbidden

NOTE status code → describes type of ~~resp~~ response status codes are like guidelines that you can alter as per requirement

eg:- 404 status code = requested piece of data not found.

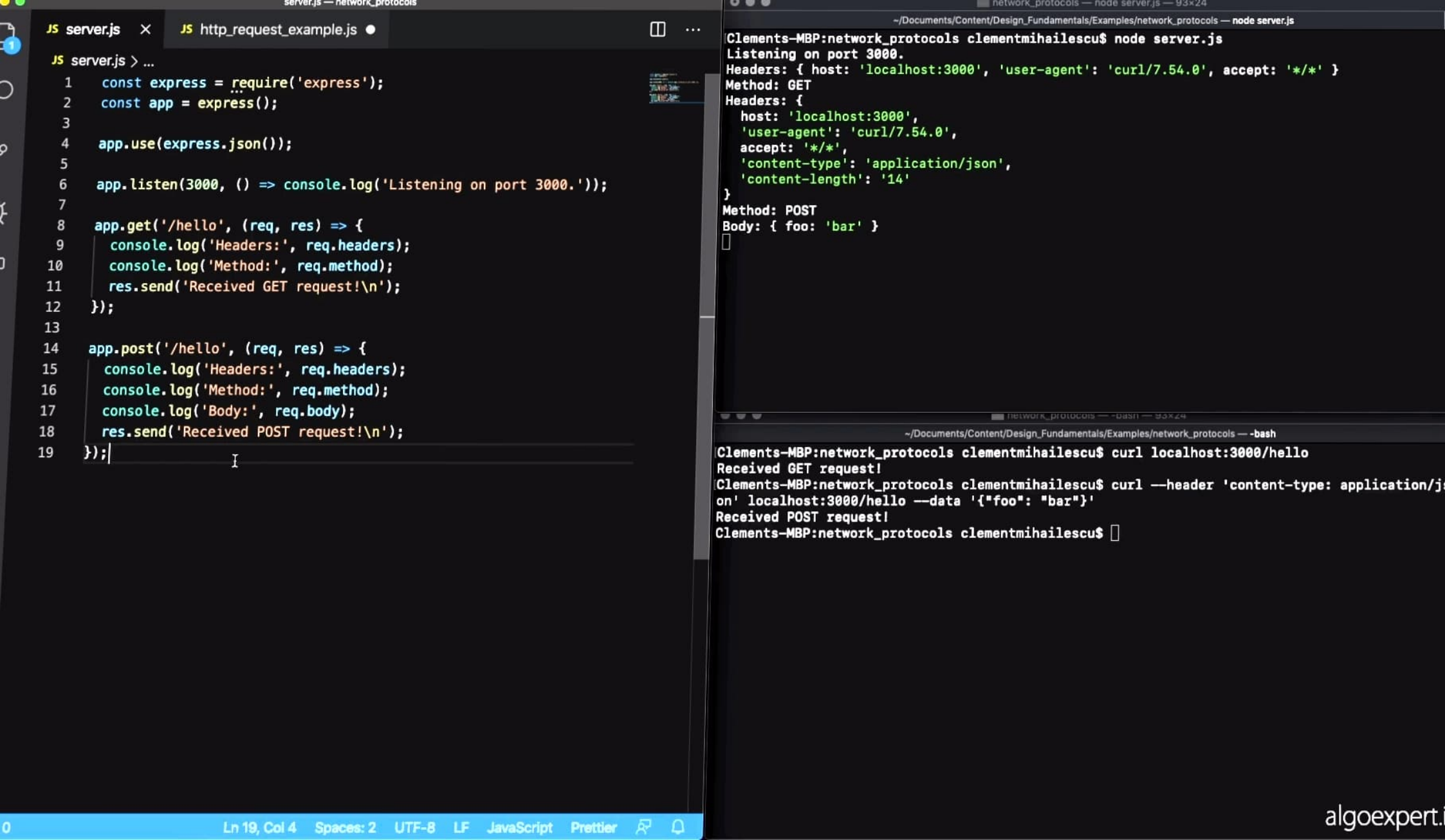
Typical HTTP req. schema

host : string (eg : facebook.com)
port : integer (eg :- 3000, 80, 43)
method : string (eg :- GET, PUT, POST, DELETE, OPTIONS, PATCH)
headers : pair-list (eg :- "Content-Type" ⇒ application/json")
body : opaque sequence of bytes

Typical HTTP resp. schema

error ↑ ↗ all ok

status code : integer (eg :- 404, 200)
headers : pair list (eg :- "Content-Length" ⇒ 1238)
body : opaque sequence of bytes


server.js — network_protocols
server.js
http_request_example.js
server.js > ...
1 const express = require('express');
2 const app = express();
3
4 app.use(express.json());
5
6 app.listen(3000, () => console.log('Listening on port 3000.'));
7
8 app.get('/hello', (req, res) => {
9 console.log('Headers:', req.headers);
10 console.log('Method:', req.method);
11 res.send('Received GET request!\n');
12 });
13
14 app.post('/hello', (req, res) => {
15 console.log('Headers:', req.headers);
16 console.log('Method:', req.method);
17 console.log('Body:', req.body);
18 res.send('Received POST request!\n');
19 });
Ln 19, Col 4
Spaces: 2
UTF-8
LF
JavaScript
Prettier
network_protocols — node server.js — 93×24
~/Documents/Content/Design_Fundamentals/Examples/network_protocols — node server.js
Clements-MBP:network_protocols clementmihailescu$ node server.js
Listening on port 3000.
Headers: { host: 'localhost:3000', 'user-agent': 'curl/7.54.0', accept: '*/*' }
Method: GET
Headers: {
host: 'localhost:3000',
'user-agent': 'curl/7.54.0',
accept: '*/*',
'content-type': 'application/json',
'content-length': '14'
}
Method: POST
Body: { foo: 'bar' }
network_protocols — -bash — 93×24
~/Documents/Content/Design_Fundamentals/Examples/network_protocols — -bash
Clements-MBP:network_protocols clementmihailescu$ curl localhost:3000/hello
Received GET request!
Clements-MBP:network_protocols clementmihailescu$ curl --header 'content-type: application/j
on' localhost:3000/hello --data '{"foo": "bar"}'
Received POST request!
Clements-MBP:network_protocols clementmihailescu$
algoexpert.