

System Design

Client-Server Model

6 Key Terms

Client

A machine or process that requests data or service from a server.

Note that a single machine or piece of software can be both a client and a server at the same time. For instance, a single machine could act as a server for end users and as a client for a database.

Server

A machine or process that provides data or service for a client, usually by listening for incoming network calls.

Note that a single machine or piece of software can be both a client and a server at the same time. For instance, a single machine could act as a server for end users and as a client for a database.

Client—Server Model

The paradigm by which modern systems are designed, which consists of clients requesting data or service from servers and servers providing data or service to clients.

IP Address

An address given to each machine connected to the public internet. IPv4 addresses consist of four numbers separated by dots: **a.b.c.d** where all four numbers are between 0 and 255. Special values include:

- **127.0.0.1**: Your own local machine. Also referred to as **localhost**.
- **192.168.x.y**: Your private network. For instance, your machine and all machines on your private wifi network will usually have the **192.168** prefix.

Port

In order for multiple programs to listen for new network connections on the same machine without colliding, they pick a **port** to listen on. A port is an integer between 0 and 65,535 (2^{16} ports total).

Typically, ports 0-1023 are reserved for *system ports* (also called *well-known* ports) and shouldn't be used by user-level processes. Certain ports have pre-defined uses, and although you usually won't be required to have them memorized, they can sometimes come in handy. Below are some examples:

- 22: Secure Shell
- 53: DNS lookup
- 80: HTTP
- 443: HTTPS

DNS

Short for Domain Name System, it describes the entities and protocols involved in the translation from domain names to IP Addresses. Typically, machines make a DNS query to a well known entity which is responsible for returning the IP address (or multiple ones) of the requested domain name in the response.

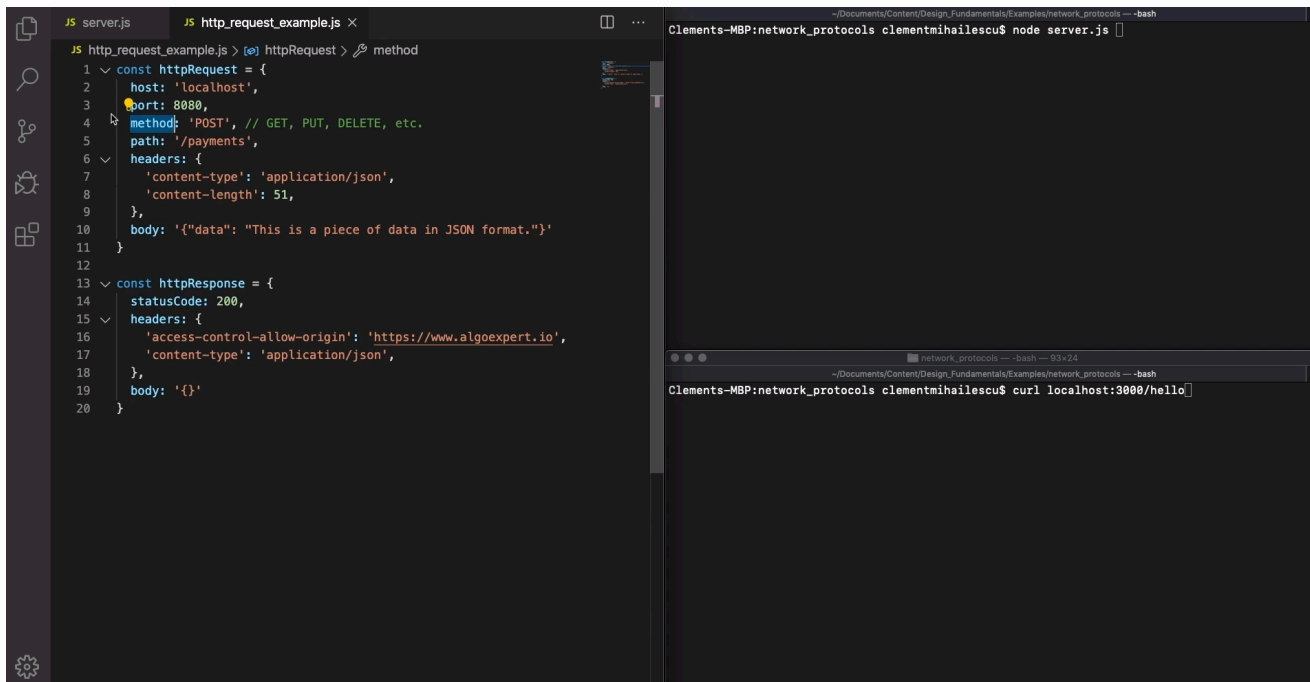
3/14/2021

Network Protocols

IP - Internet Protocols, IPv4 & IPv6. IP packets contain header (source IP address, destination IP address) and IP data (stored in 2^{16} bytes)

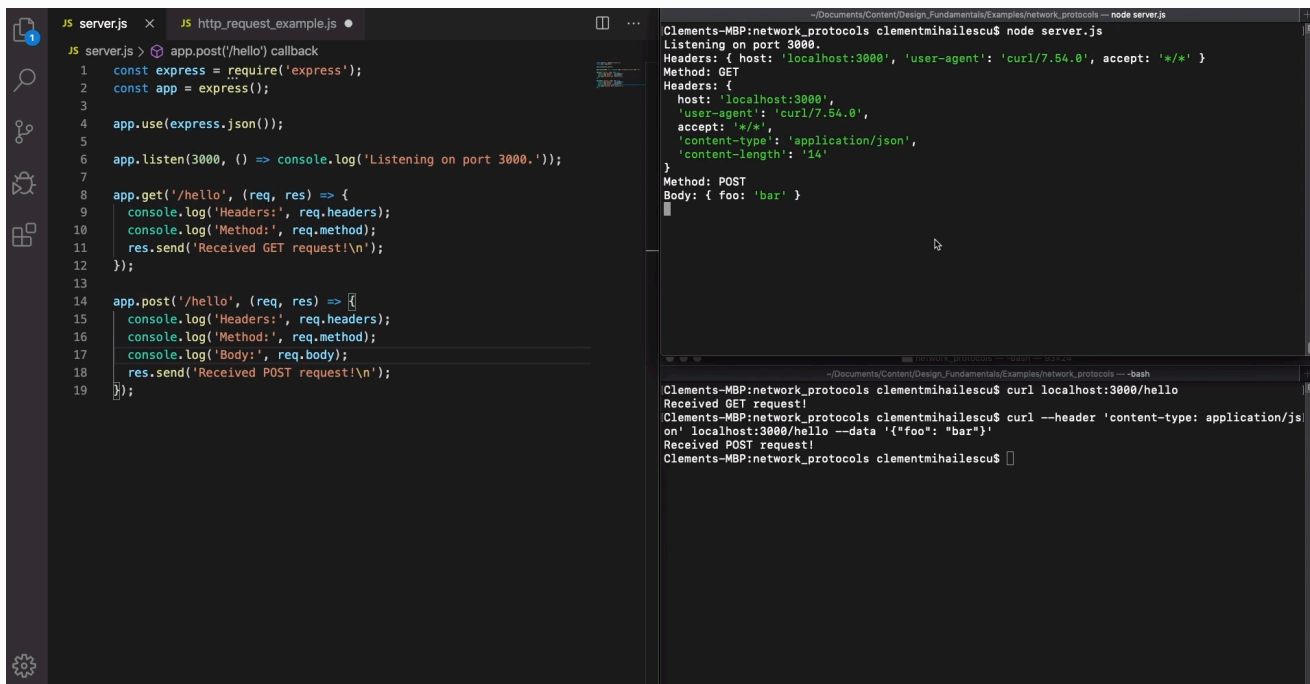
TCP - Transmission and Control Protocols. Arbitrary long data in order. TCP data in IP data.

HTTP - create TCP connections/interactions/handshake to define meaningful information by requests and responses (objects)



```
JS http_request_example.js > http request > method
1 const http request = {
2   host: 'localhost',
3   port: 8080,
4   method: 'POST', // GET, PUT, DELETE, etc.
5   path: '/payments',
6   headers: {
7     'content-type': 'application/json',
8     'content-length': 51,
9   },
10  body: '{"data": "This is a piece of data in JSON format."}'
11 }
12
13 const http response = {
14   statusCode: 200,
15   headers: {
16     'access-control-allow-origin': 'https://www.algoexpert.io',
17     'content-type': 'application/json',
18   },
19   body: {}
20 }
```

```
Clements-MBP:network_protocols clementmihailescu$ node server.js
```



```
JS server.js > app.post('/hello') callback
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 });
```

```
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' }
```

```
Clements-MBP:network_protocols clementmihailescu$ curl localhost:3000/hello
Received GET request!
Clements-MBP:network_protocols clementmihailescu$ curl --header 'content-type: application/json' localhost:3000/hello --data '{"foo": "bar"}'
Received POST request!
Clements-MBP:network_protocols clementmihailescu$
```

4 Key Terms

IP

Stands for **Internet Protocol**. This network protocol outlines how almost all machine-to-machine communications should happen in the world. Other protocols like **TCP**, **UDP** and **HTTP** are built on top of IP.

TCP

Network protocol built on top of the Internet Protocol (IP). Allows for ordered, reliable data delivery between machines over the public internet by creating a **connection**.

TCP is usually implemented in the kernel, which exposes **sockets** to applications that they can use to stream data through an open connection.

HTTP

The **HyperText Transfer Protocol** is a very common network protocol implemented on top of TCP. Clients make HTTP requests, and servers respond with a response.

Requests typically have the following schema:

```
host: string (example: algoexpert.io)
port: integer (example: 80 or 443)
method: string (example: GET, PUT, POST, DELETE, OPTIONS or PATCH)
headers: pair list (example: "Content-Type" => "application/json")
body: opaque sequence of bytes
```

Responses typically have the following schema:

```
status code: integer (example: 200, 401)
headers: pair list (example: "Content-Length" => 1238)
body: opaque sequence of bytes
```

IP Packet

Sometimes more broadly referred to as just a (network) **packet**, an IP packet is effectively the smallest unit used to describe data being sent over **IP**, aside from bytes. An IP packet consists of:

- an **IP header**, which contains the source and destination **IP addresses** as well as other information related to the network
- a **payload**, which is just the data being sent over the network