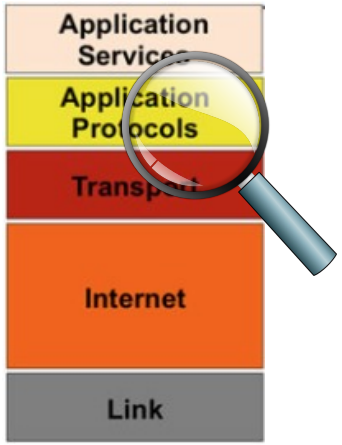
Ch. 13 - IoT Application Protocol Layer

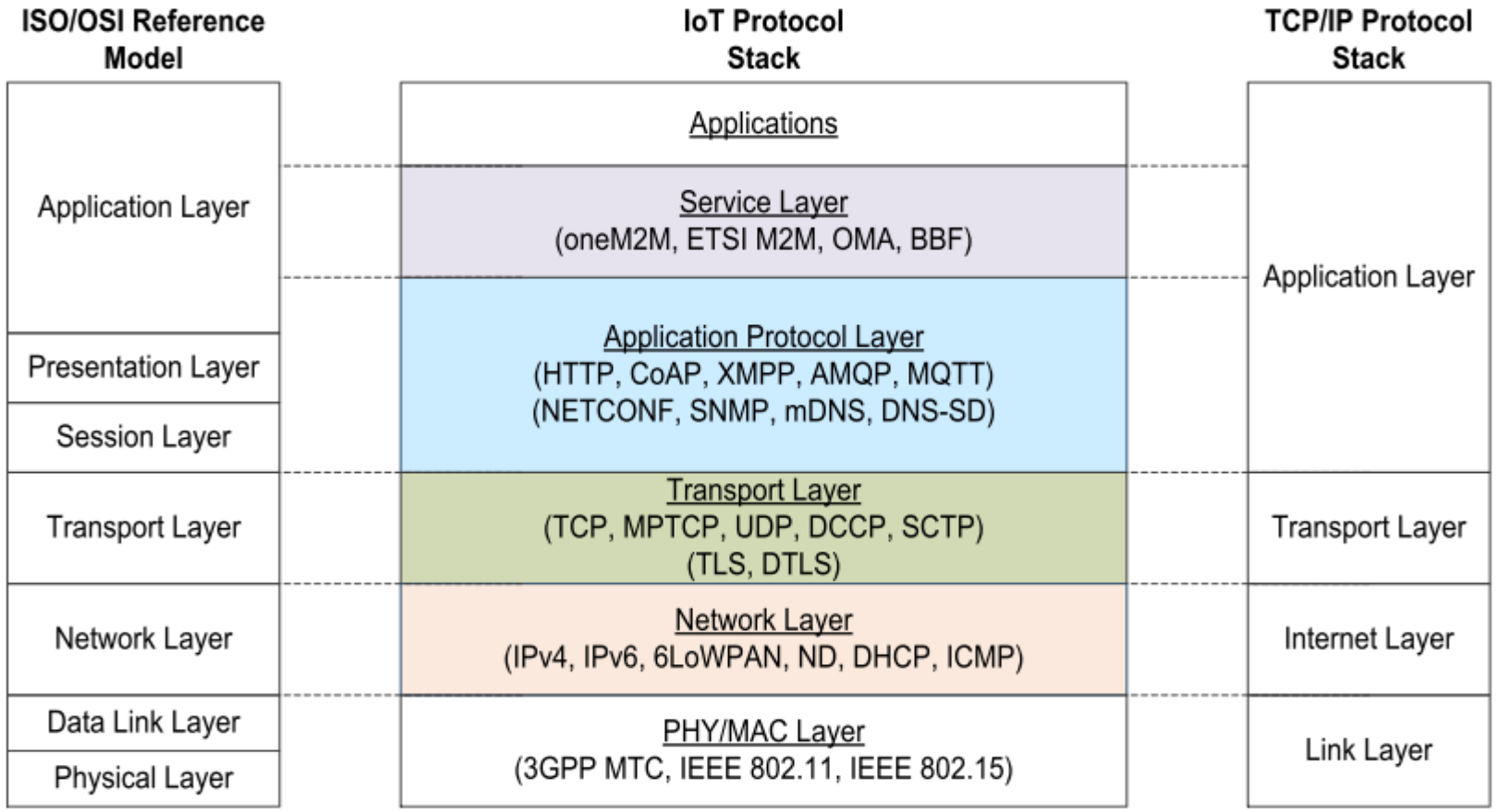
Sec 1 – HTTP Protocol

COMPSCI 147

Internet-of-Things; Software and Systems



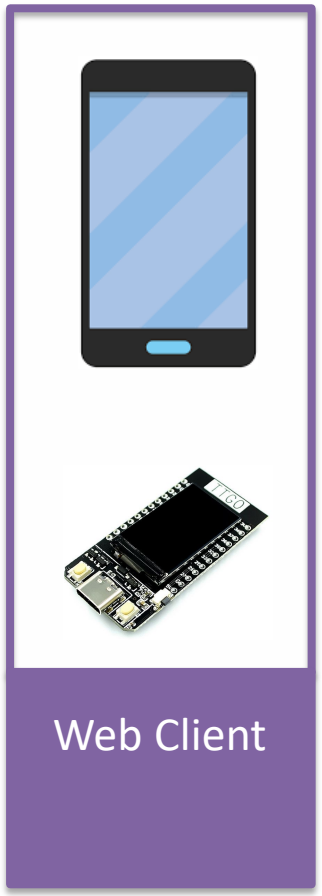
IOT PROTOCOL STACK: A LAYERED VIEW



APP PROTOCOLS FOR IOT - STANDARDIZATION

- **HTTP**
  - **IETF standard (RFC 2616 is HTTP/1.1)**
- **CoAP**
  - **IETF standard (RFC 7252)**
- **XMPP**
  - **IETF standard (RFC 6272)**
- **MQTT**
  - **OASIS standard**
- AMQP
  - OASIS and ISO 19464 standard (1.0)
- SIP
  - IETF Standard (RFC 3261)
- IEEE 1888
  - IEEE Standard
- DDS (RTPS)
  - Object Management Group (OMG) Standard

Recall: What is a web-server

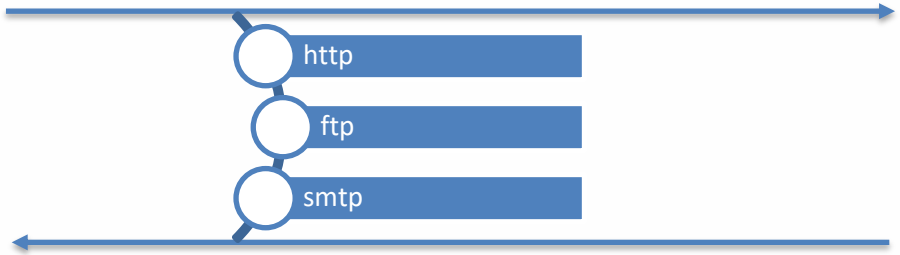


Web Client

Web Server
A network application listening on a port.

Protocols

Web Client



A set of rules that allow two or more entities of a communication system to transmit information.

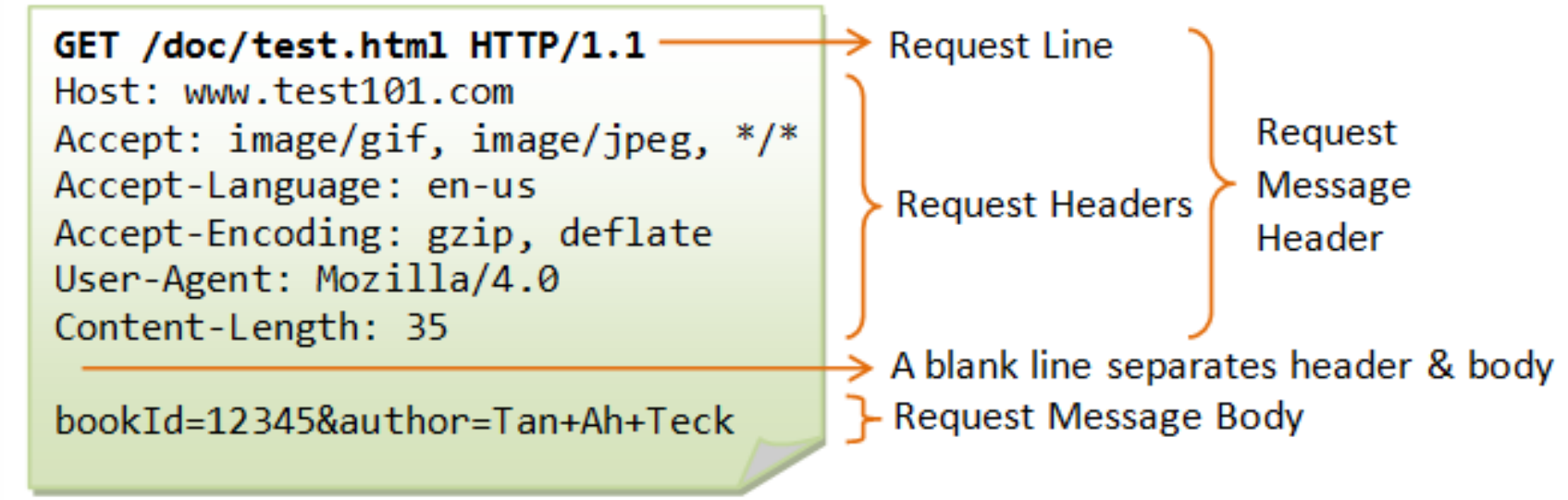
Syntax, semantics, and synchronization of communication and possible error recovery methods between communication systems are all defined by the term “protocol”.

Web Server

A network application listening on a port.

http

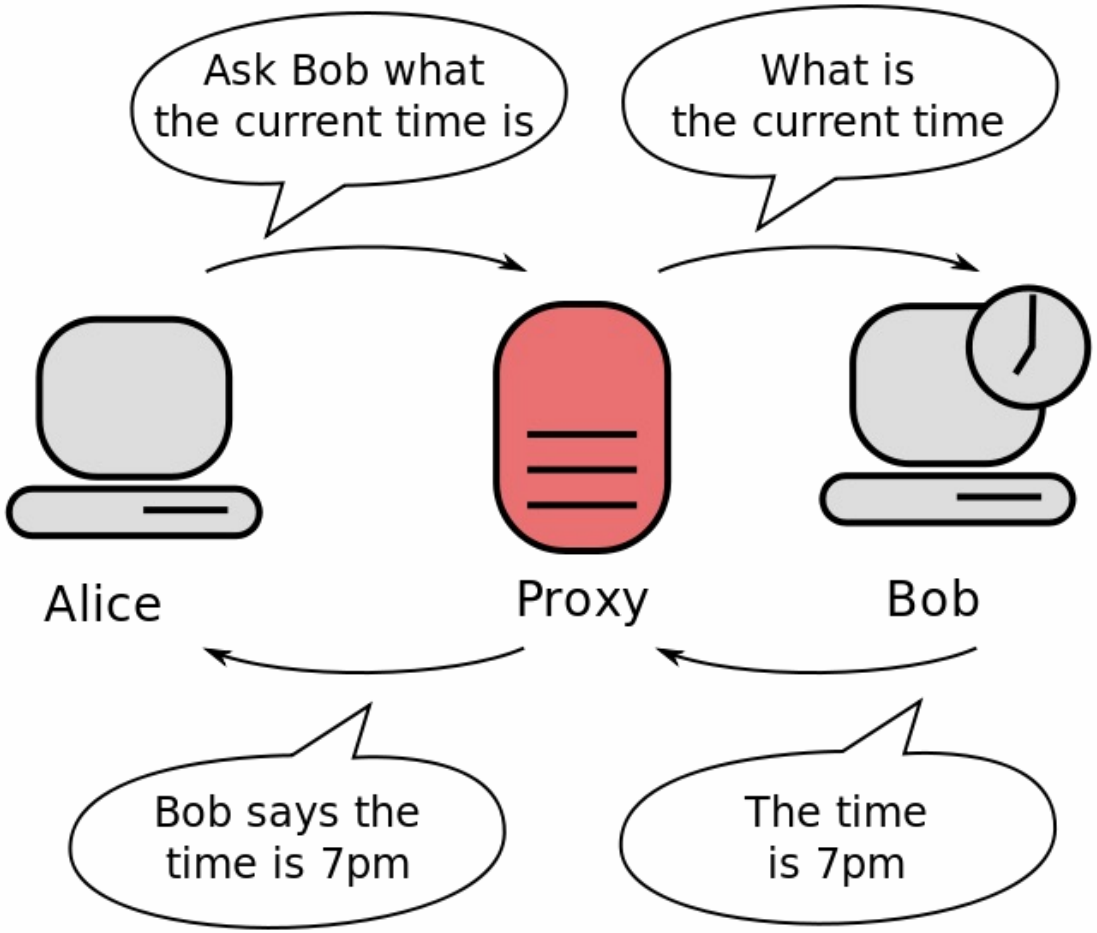
- Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web
- Load web pages using hypertext:
  i.e., text with (hyper)links to other text



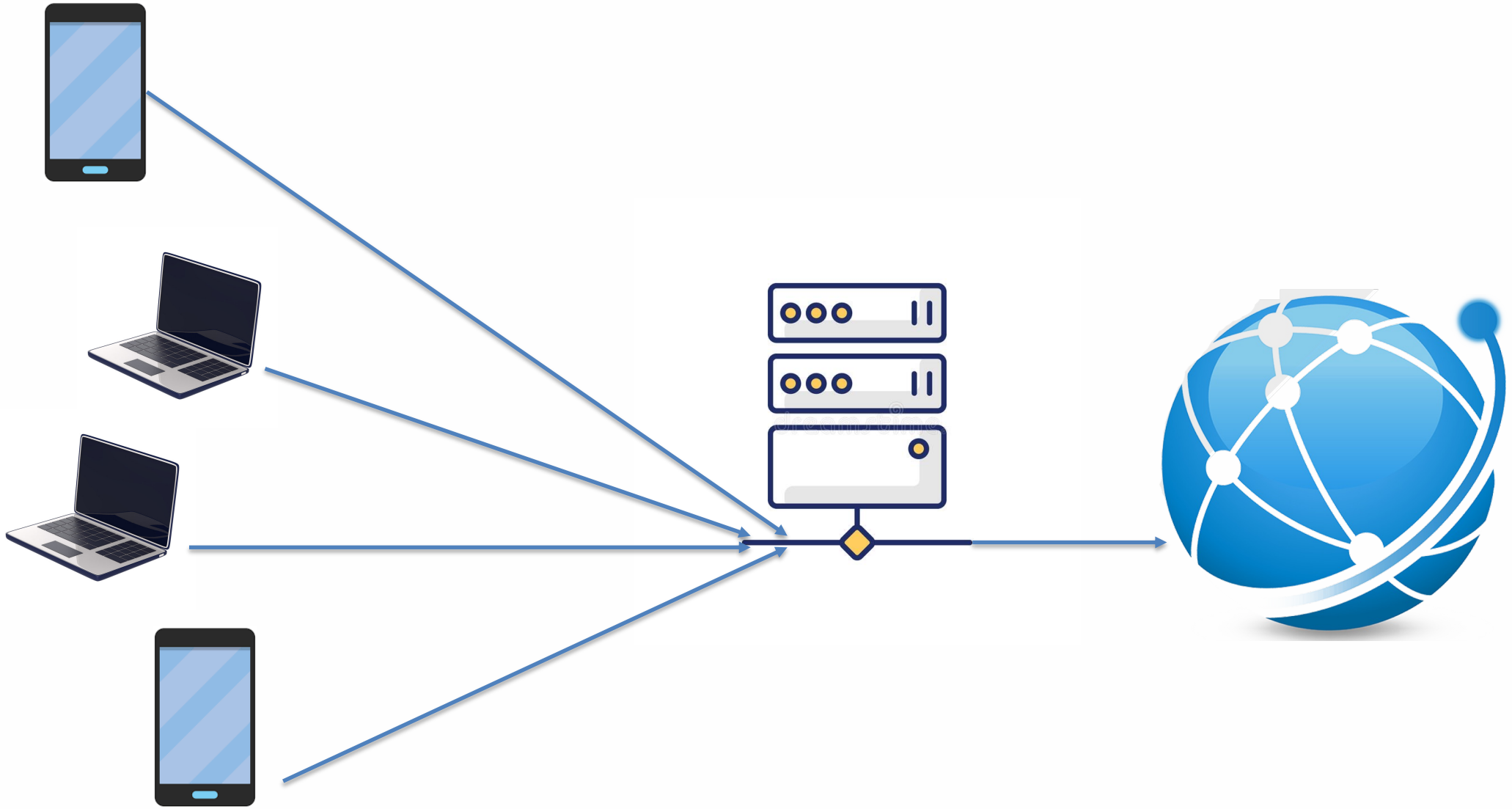
Goal: Suppose you are a system administrator at UCI, and you want to perform some of the following:

- Before the exam, all students in the same class tend to search similar queries. We want to cache them temporarily to not request the same page and again from the web.
- You also want to inspect the raw request and block malicious/insecure websites.
- You want to log website activity (while preserving privacy by anonymizing PII)

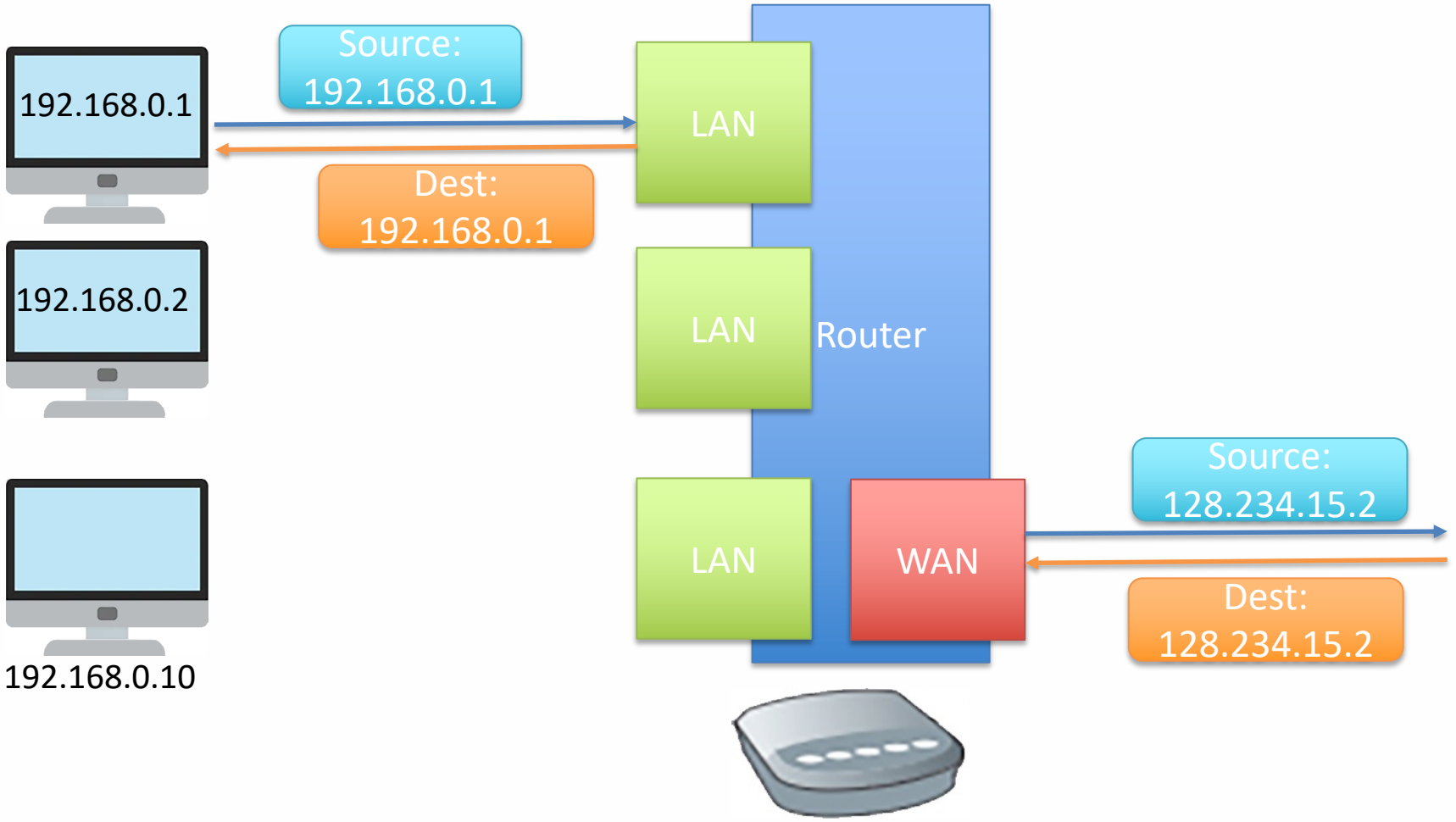
Proxy server



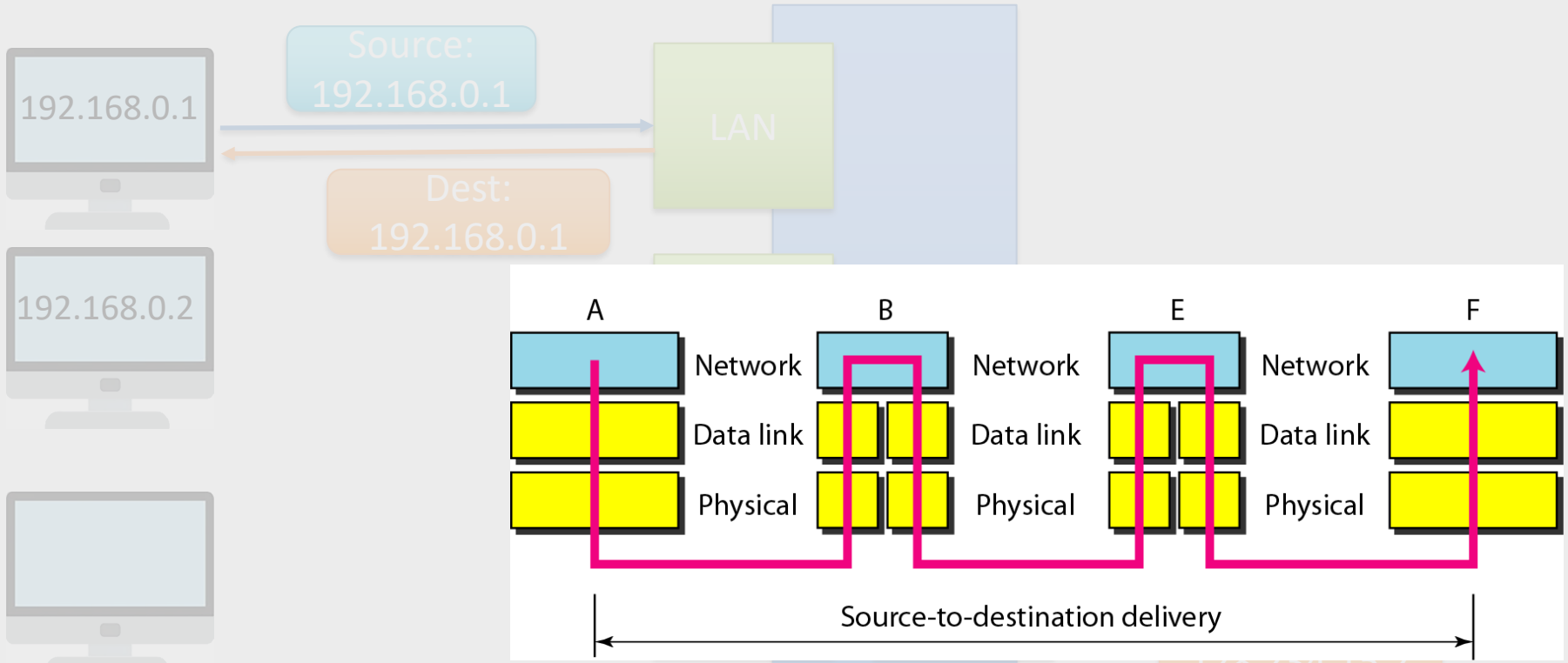
Proxy server



What is the difference between a proxy server and a router ?

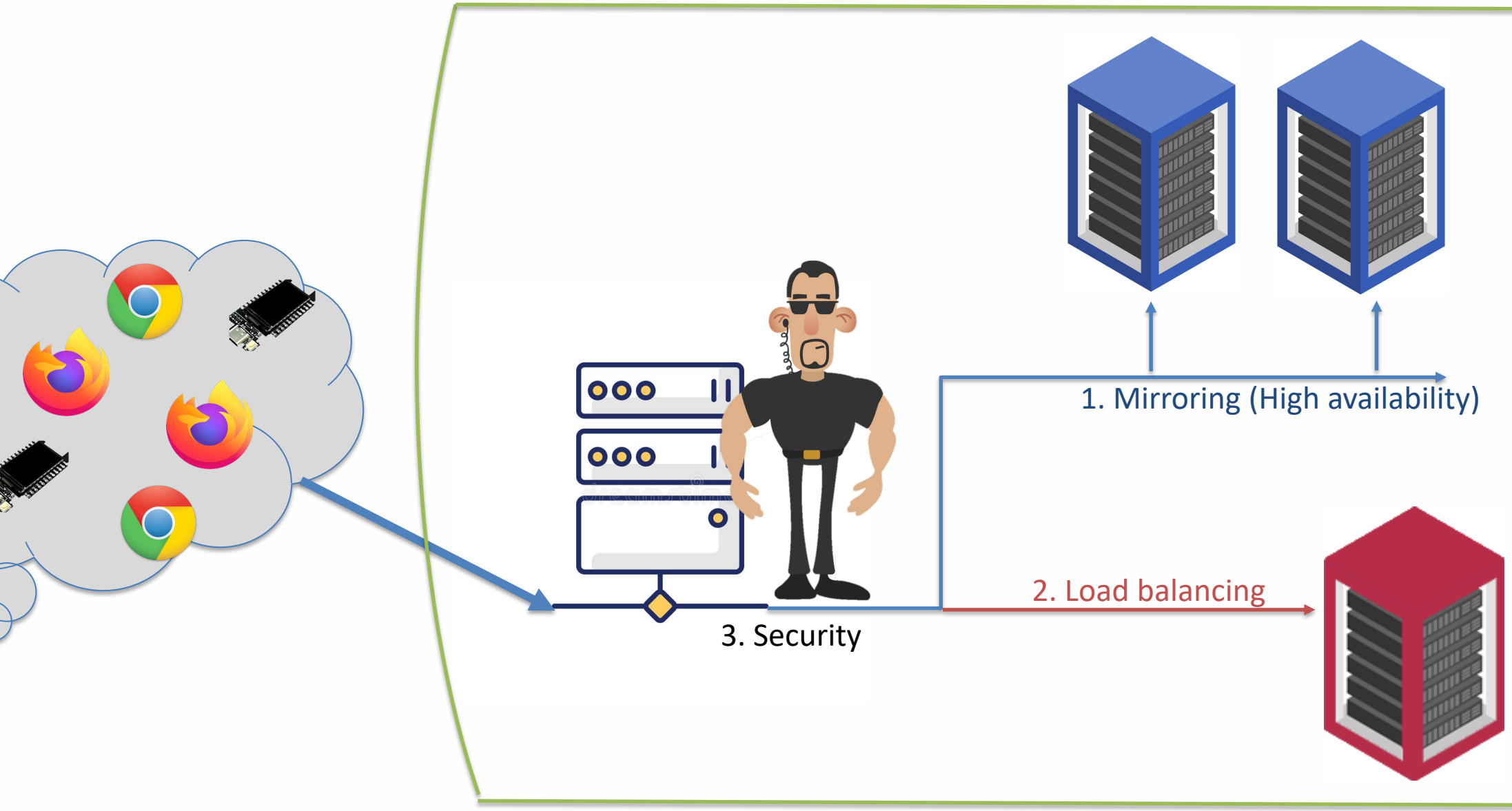


What is the difference between a proxy server and a router ?



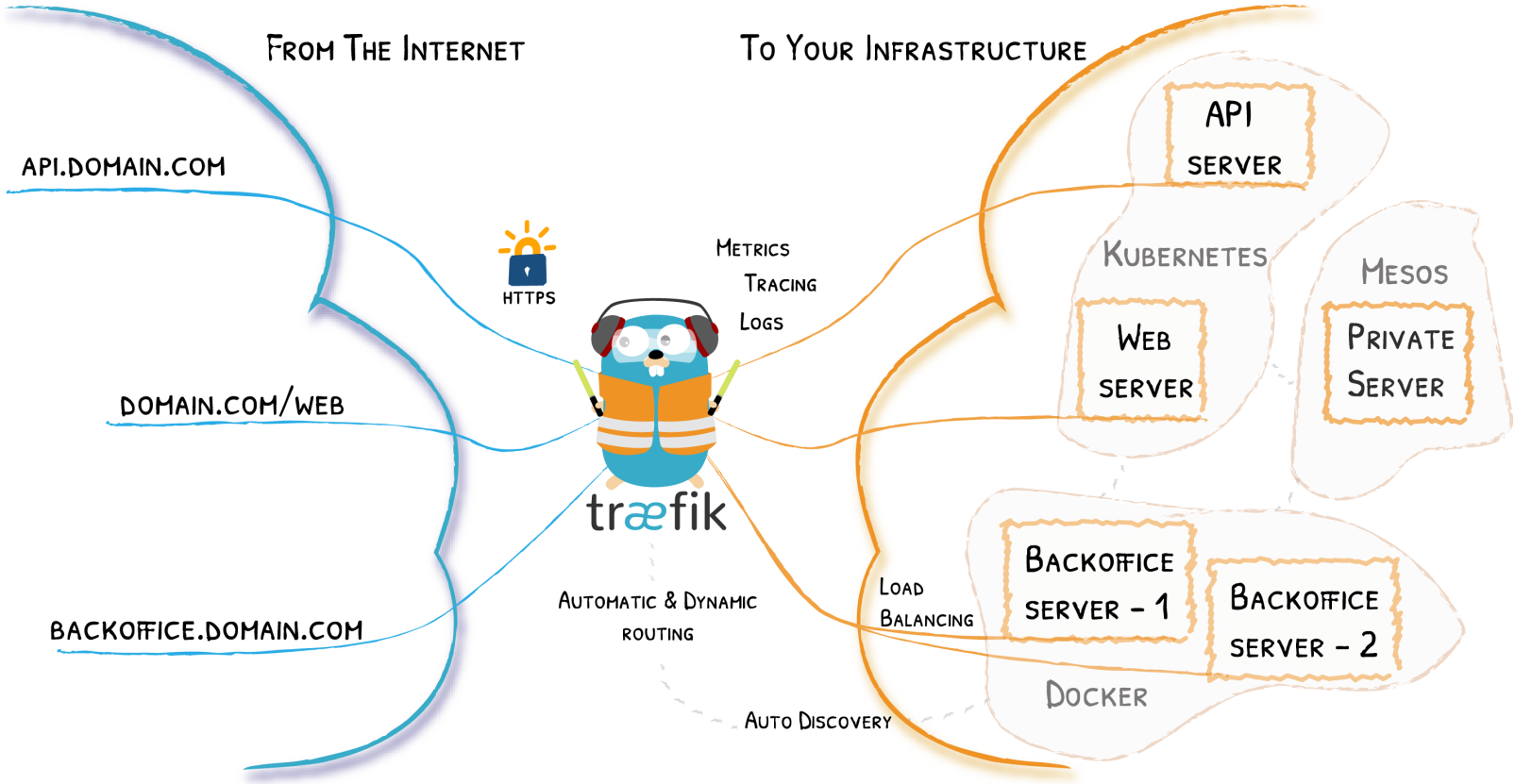
Hint

What is a reverse proxy ?

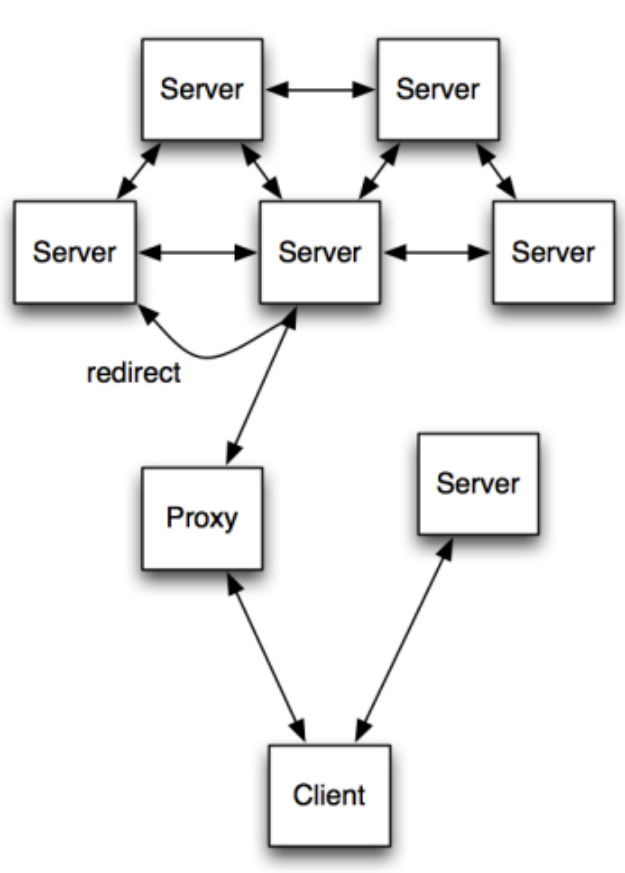


Examples of reverse proxy

- Nginx reverse proxy
- Traefik

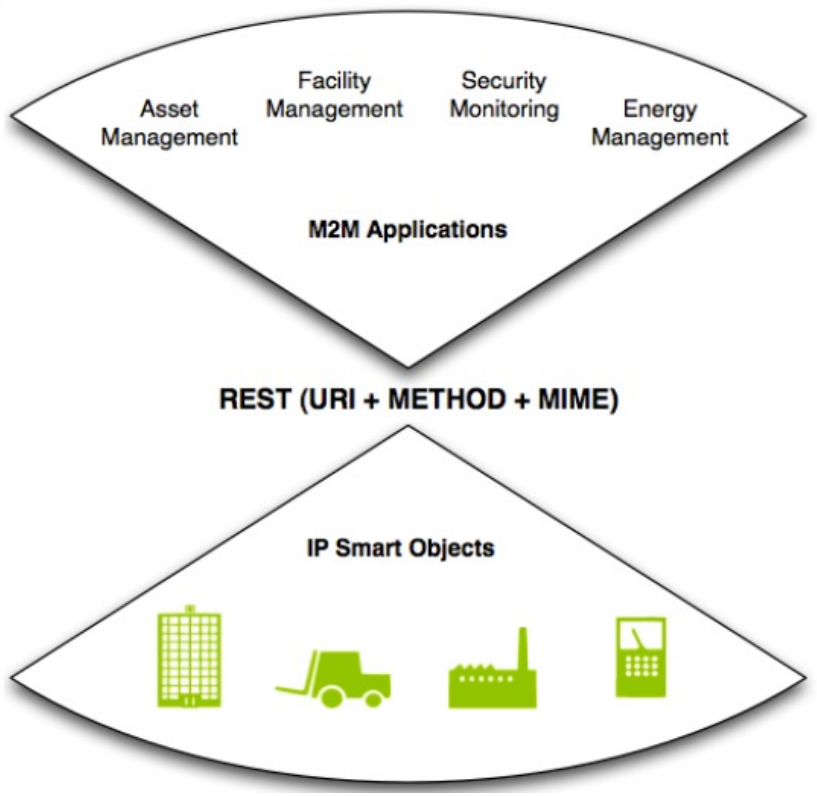


HTTP - THE WEB ARCHITECTURE THAT CAN SCALE UP



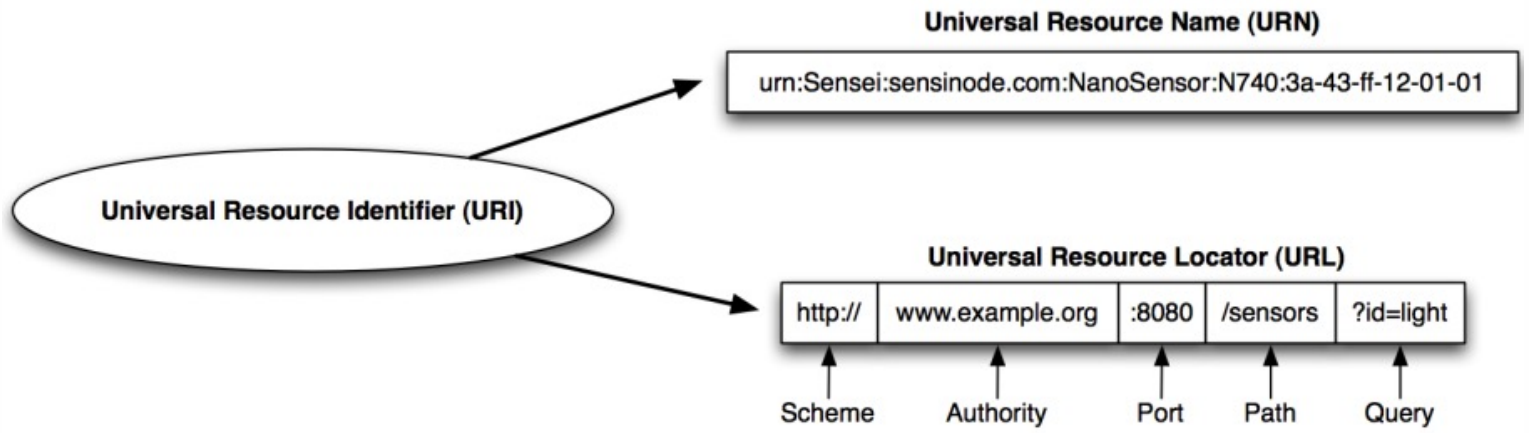
HTTP - THE WEB OF THINGS

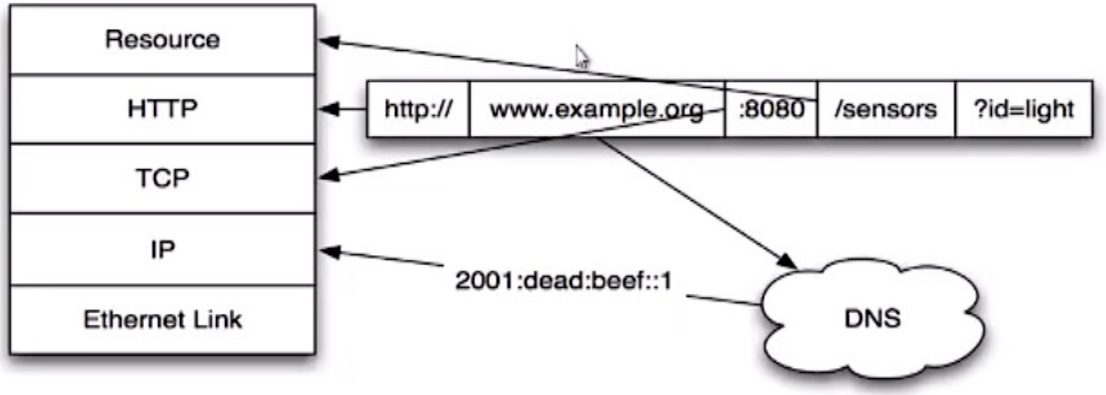
- Powering M2M with a **Web of Things**



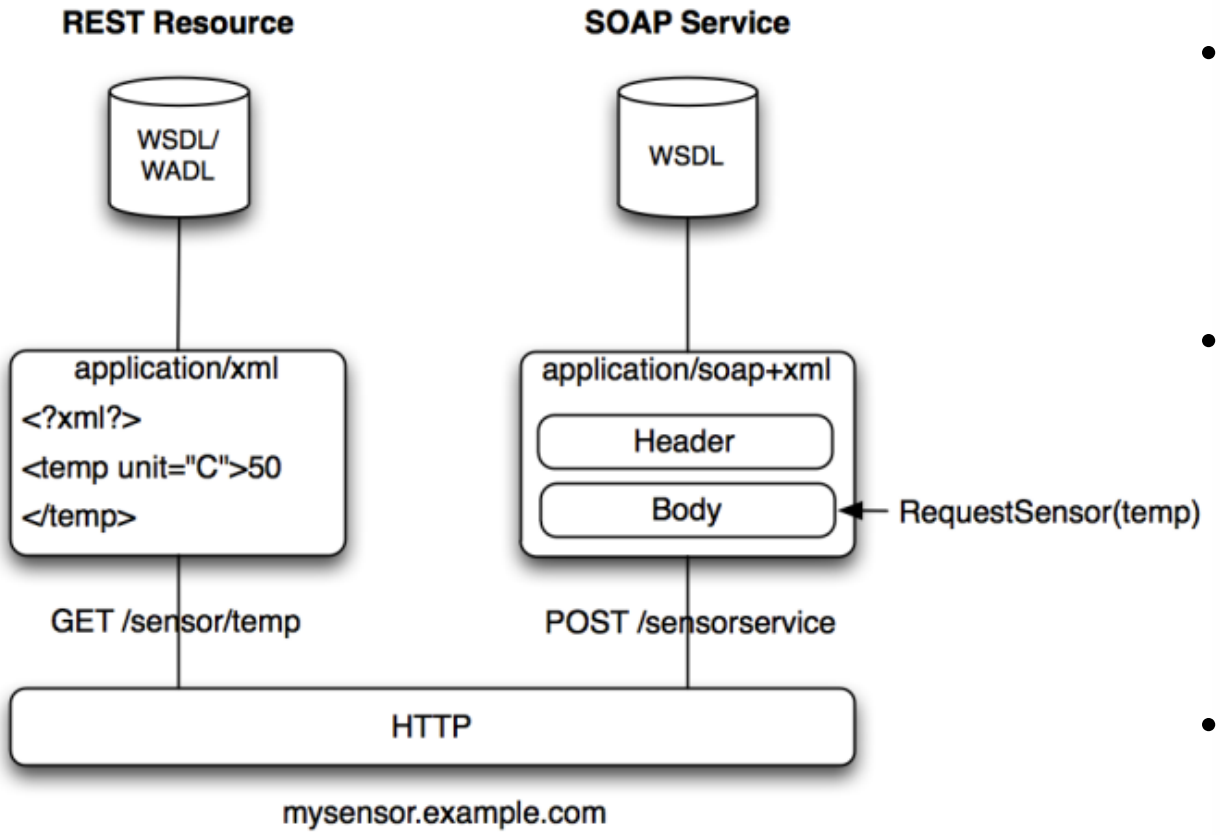
- Representational State Transfer (REST)
  - Used to create standard interfaces among devices
  - Concept / Blueprint
- Rest consist of:
  - Unified Resource Identifier
  - HTTP Methods
    - https://restfulapi.net/http-methods/
    - HTTP GET
    - HTTP POST
    - HTTP PUT
    - HTTP DELETE
    - HTTP PATCH
  - Multipurpose Internet Mail Extension

HTTP - WEB RESOURCE IDENTIFICATION





THE WEB SERVICE PARADIGM – SOAP AND REST

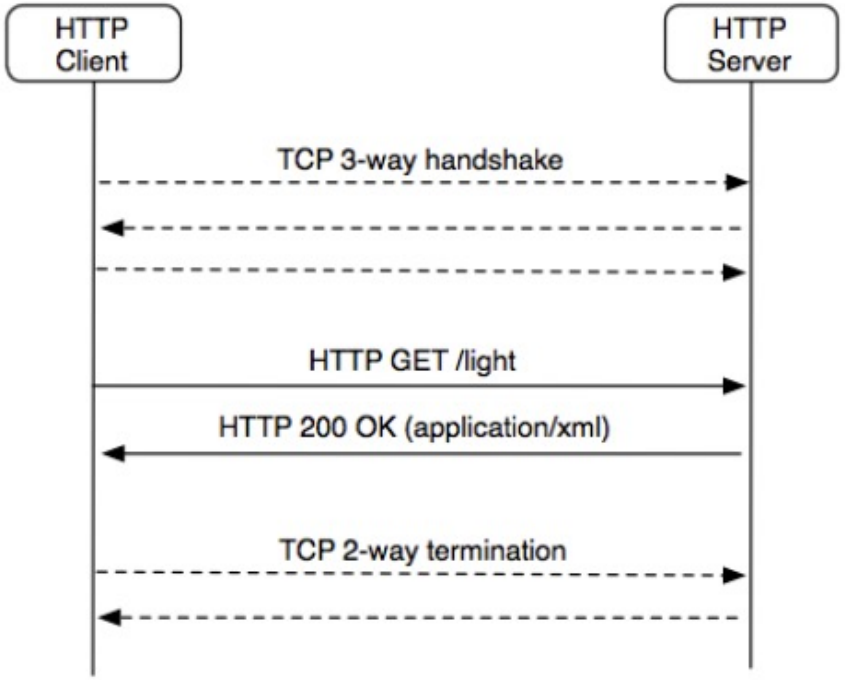


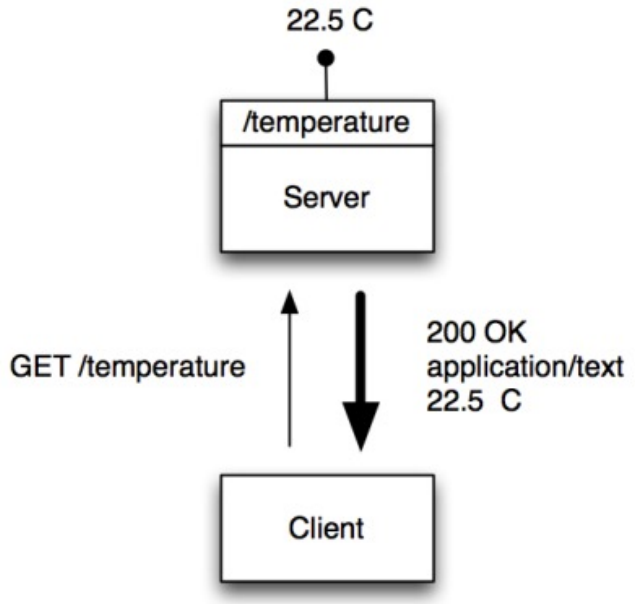
- SOAP *(Microsoft, 1998)*
  - Simple Object Access Protocol
  - XML based
- REST *(UCI student's Doctoral Thesis, 2000)*
  - Representational state transfer
  - URL/HTTP1.1 based

- WSDL (Web Services Description Language)
  - XML language for describing web services.
- WADL (Web Application Description Language)
  - XML description of HTTP-based web services

AN HTTP REQUEST

- HTTP is not very flexible for IoT applications
  - Lot of communication if you just need to send a simple message
  - You need to keep connection opens, time outs, etc..



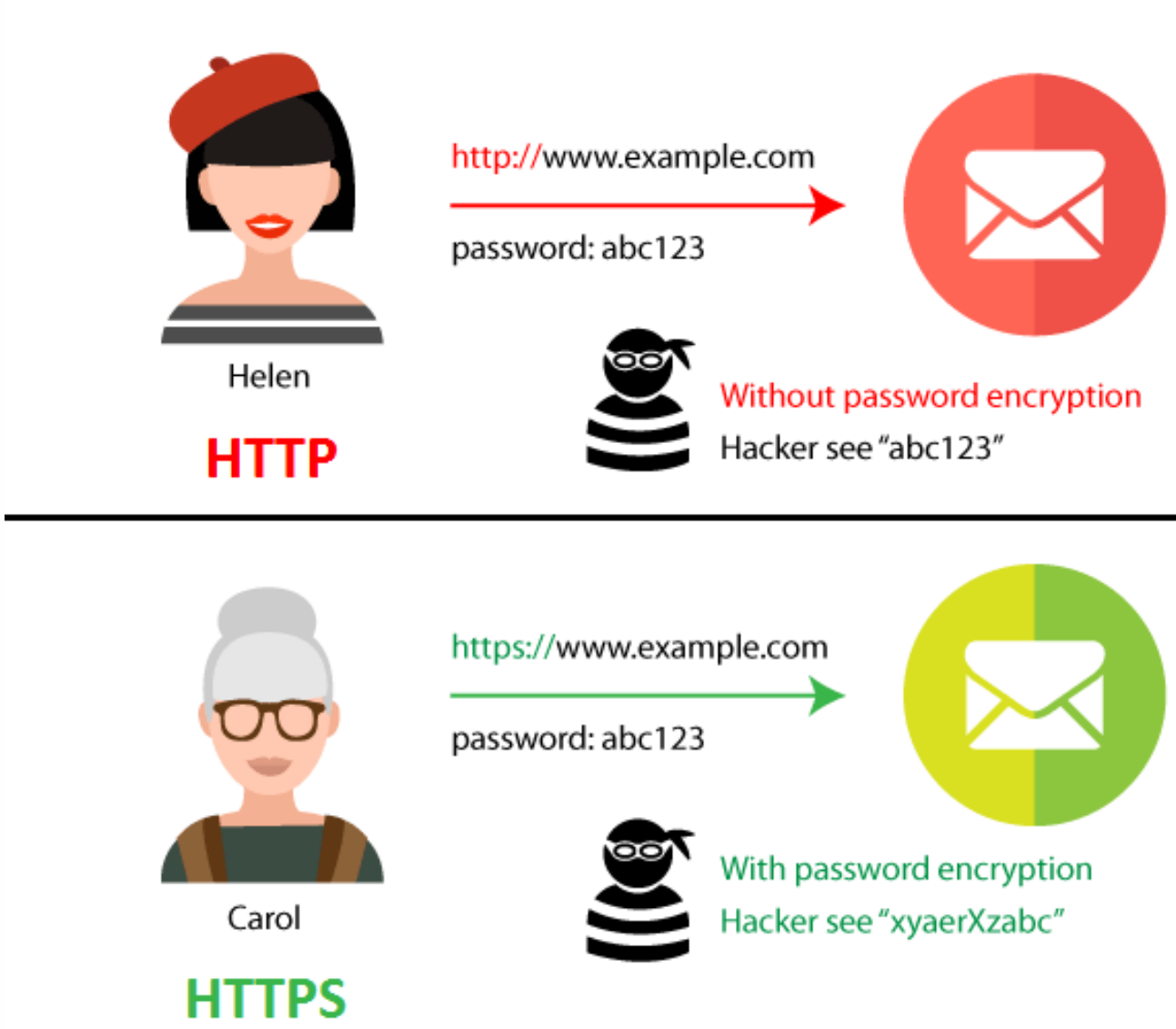


Security in http

How to secure http

- In standard HTTP, all information is sent in clear text
- Vulnerable to hackers who are snooping on the internet

How to secure http



Under the hood: https using ssl

Step 1: Buy a domain

Step 2: Contact certificate authority (CA) to prove you are the owner of the domain

Step 3: Server receives a ssl certificate that contains the public key of the server but encrypted with CA's private key.

Step 4: When a client tries to connect, server sends it's ssl certificate

Step 5: Client uses CA's public key in browser certificate store to decode public key of server

Step 6: Client now encrypts all messages with public key of server and sends it over internet.

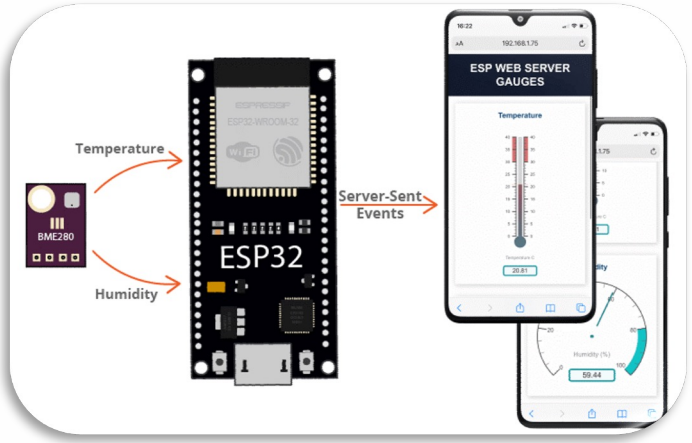
https://www.youtube.com/watch?v=AQDCe585Lnc

How to develop your own server

Tying things together: how to design your http server

- 1. Choosing the right software stack
  - E.g., Flask (Python based)
  - NodeJS (JavaScript based)
  - Php …
- 2. Choosing the right software stack

Tying things together: how to design your http server



1. On embedded systems

3. On cloud server

2. On Self hosted server

Tying things together: how to design your http server

	ESP32
Number of cores	2
Processor Frequency	80~240 MHz
Memory Size	320 KiB RAM,
Storage	448 KiB ROM (Flash)
Network B/W	150 Mbps Wi-Fi: 802.11 b/g/n
Power	100mA
Cost	10$

Tying things together: how to design your http server

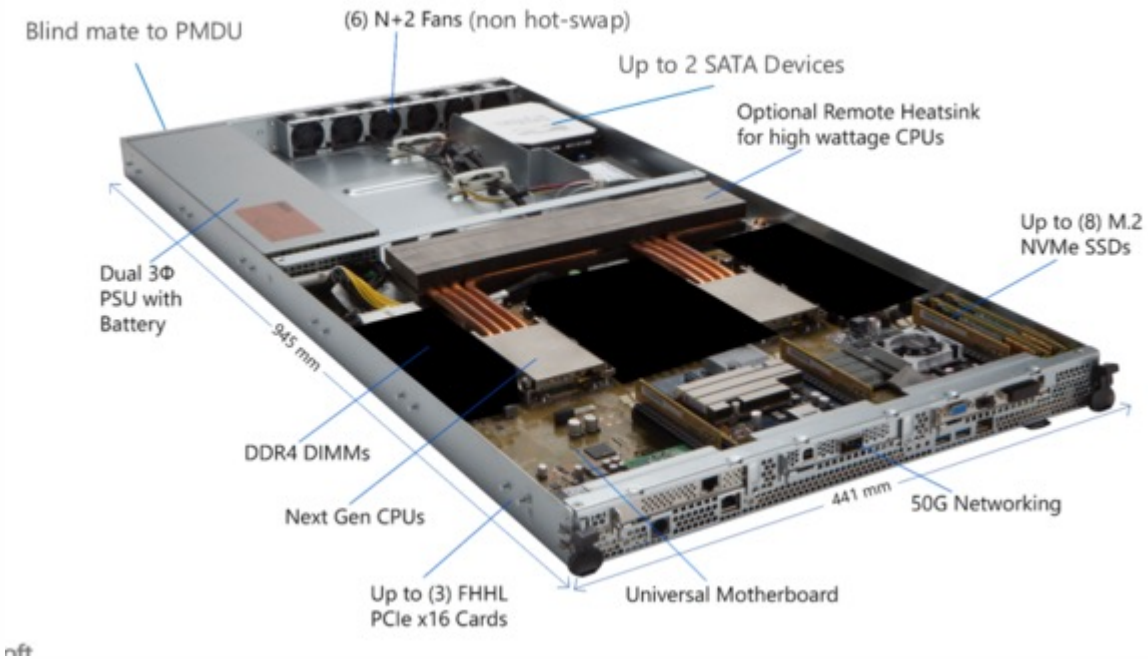
	ESP32	Self-hosted (e.g., RPi)
Number of cores	2	4
Processor Frequency	80~240 MHz	1.2 GHz
Memory Size	320 KiB RAM,	8GB
Storage	448 KiB ROM (Flash)	32 GB (SD-card)
Network B/W	150 Mbps Wi-Fi: 802.11 b/g/n	1Gbps
Power	100mA	1A
Cost	10$	100$

Tying things together: how to design your http server

	ESP32	Self-hosted (e.g., RPi)	Cloud
Number of cores	2	4	?
Processor Frequency	80~240 MHz	1.2 GHz	?
Memory Size	320 KiB RAM,	8GB	?
Storage	448 KiB ROM (Flash)	32 GB (SD-card)	?
Network B/W	150 Mbps Wi-Fi: 802.11 b/g/n	1Gbps	?
Power	100mA	1A	?
Cost	10$	100$	?/hr

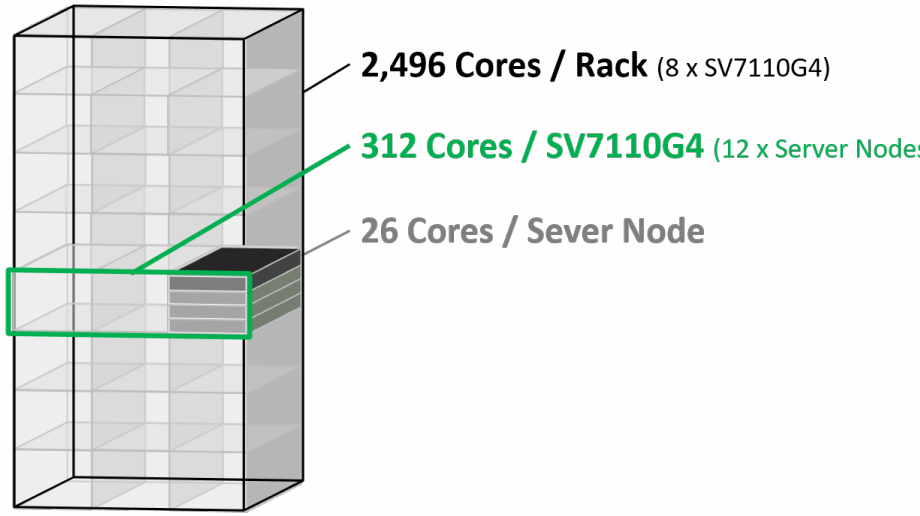
Example cloud hardware

- OpenComputeProject



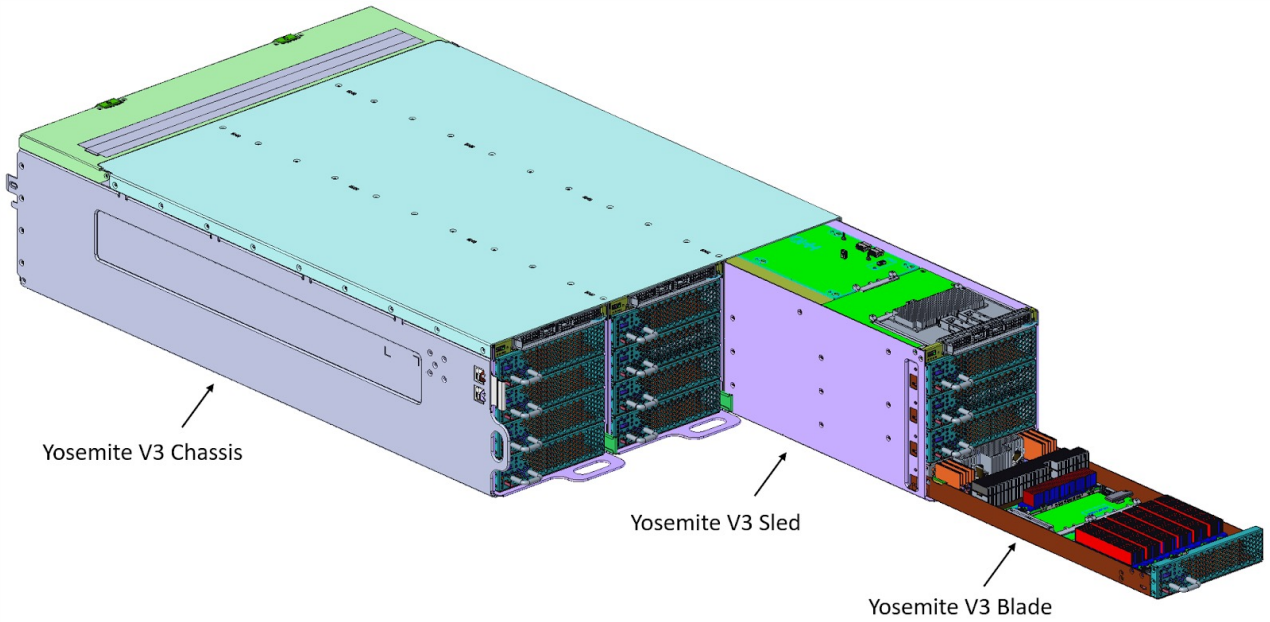
Example cloud hardware

- Example: YosemiteV3 (currently deployed in Meta datacenters)



Example cloud hardware

- Demo of Yosemite V3 hardware



Tying things together: how to design your http server

	ESP32	Self-hosted (e.g., RPi)	Cloud
Number of cores	2	4	100
Processor Frequency	80~240 MHz	1.2 GHz	4GHz
Memory Size	320 KiB RAM,	8GB	100GB
Storage	448 KiB ROM (Flash)	32 GB (SD-card)	Several TB+
Network B/W	150 Mbps Wi-Fi: 802.11 b/g/n	1Gbps	100Gbps
Power	100mA	1A	10A
Cost	10$	100$	10,000$

Questions