DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

1222·2022
800
ANNI

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

SIGNET
signet.dei.unipd.it

Introduction: technologies

ICT for Industrial Applications
(ICT4IA)

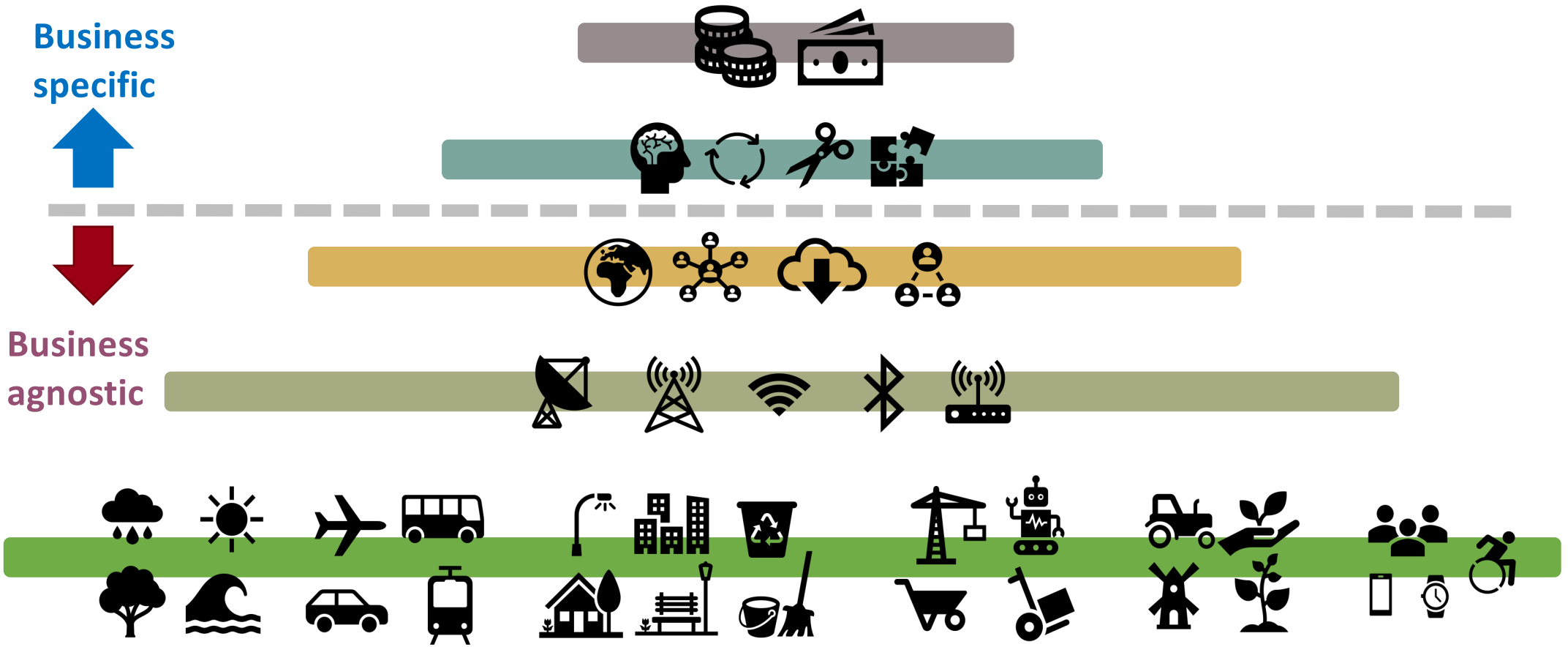
Andrea Zanella (zanella@dei.unipd.it)
Office number: (049 827)7770

SIGNALS AND NETWORKING RESEARCH GROUP

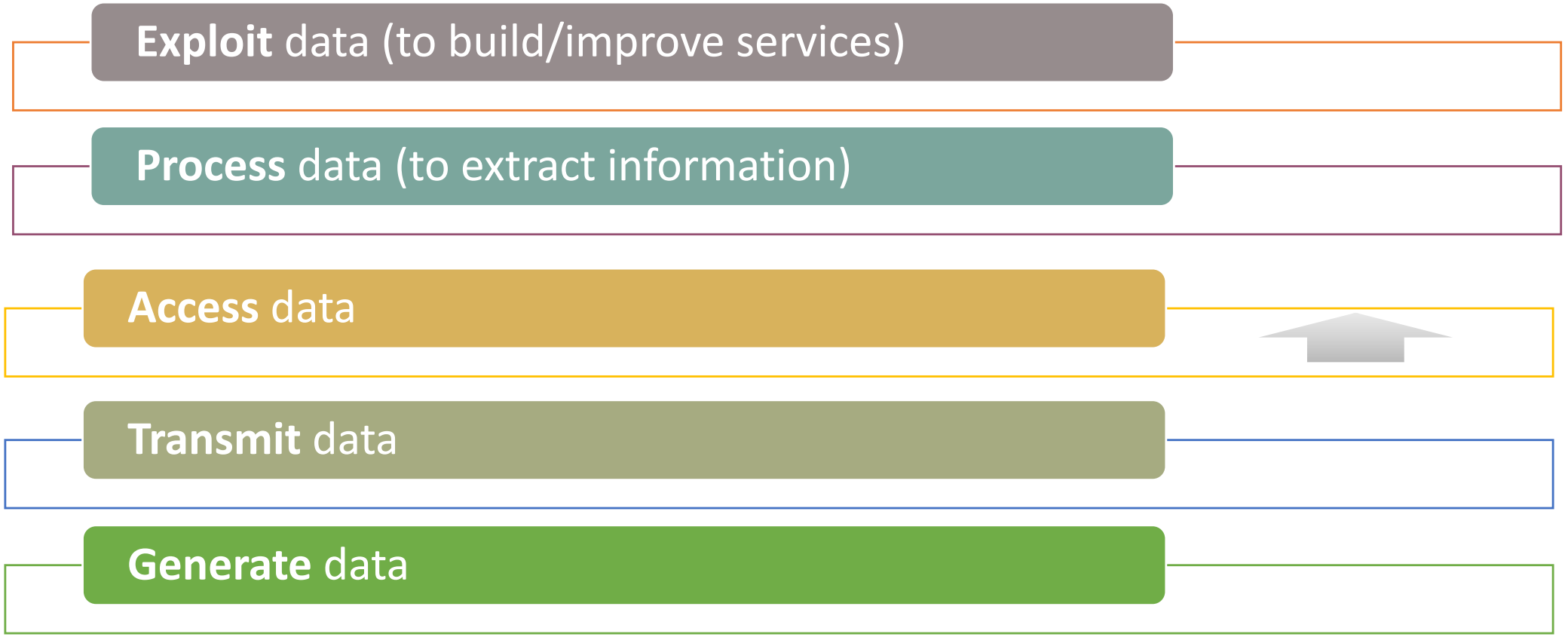
«Internet of things» what's that?

Mainly… a **change of perspective** in the development of services, systems and business models…

The IoT vision: horizontal structure



The five building blocks of IoT



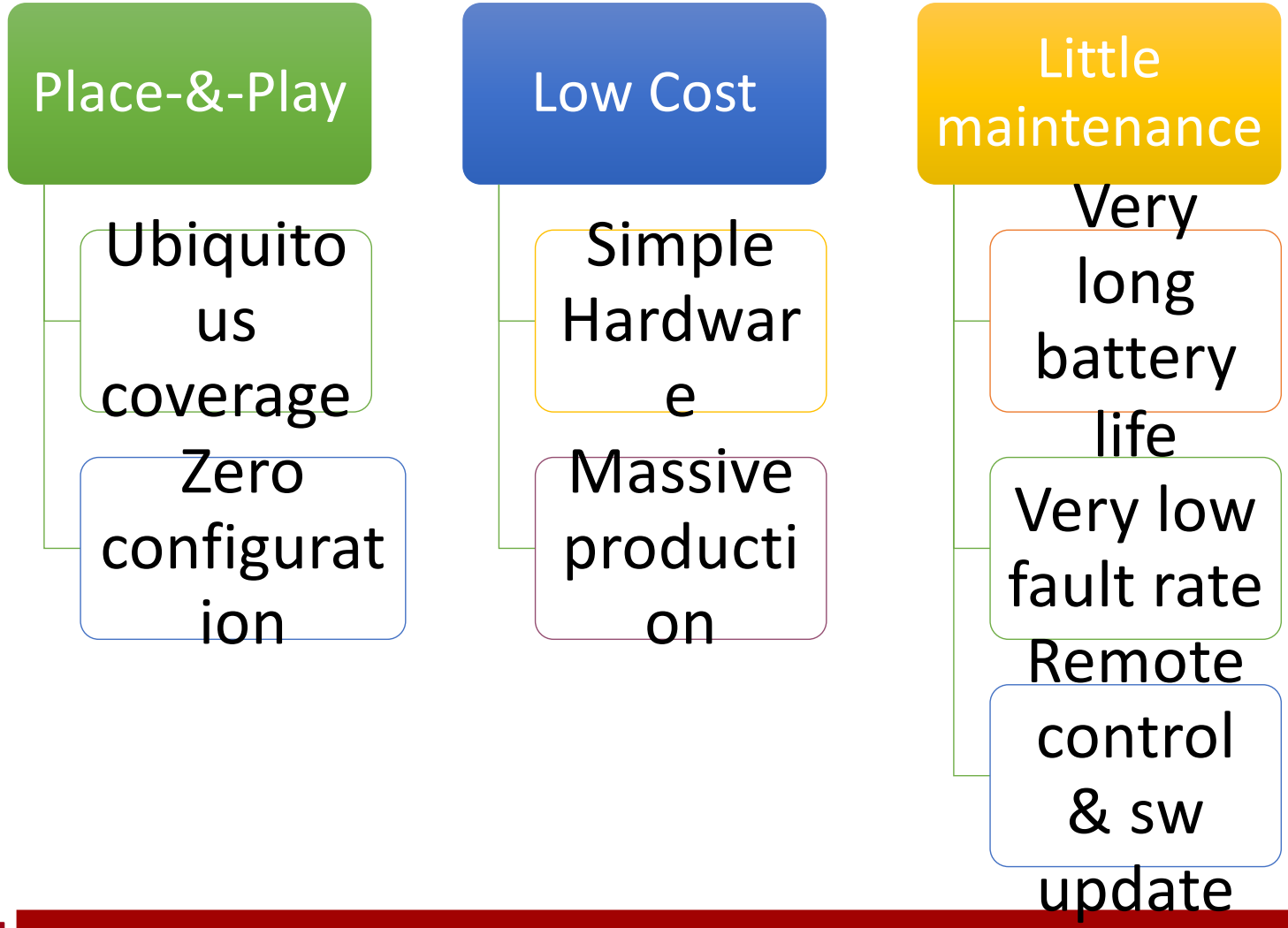
Generate data

- **Objective:** monitoring, measurement, tracking (environmental parameters, remote sensing, industrial sensors, transport, home automation, agriculture, ...)
- **Technologies:** sensors of light, temperature, humidity, pollution, presence, proximity, acceleration, pressure, power consumption, but also portable devices such as smartphones, smartwatch, wearable,...
- **Requirements:** high energy efficiency, possibility to obtain energy from the environment (energy harvesting), low cost, small size, high robustness, standard interface

Transmit data

- **Objective:** to collect data from peripheral devices
- **Technologies:** sensor data transmission technologies (Bluetooth LE, WiFi, ZigBee, Zwave, LPWAN, NB-IoT...), gateway for TCP/IP network interconnection
- **Requirements:** high energy efficiency, ubiquitous coverage, low cost, low complexity of network management, security

IoT Service Requirements



Service types

Stand alone

- Short range
- LoRaWAN

Platform-as-a-service

- LTE, NB-IoT, GSM, ...
- SigFox
- LoRaWAN

Who is the winner?

- Complementary technologies for different services
- Very likely we will need all of them
- Integration MUST occur at upper layers

Access data

- **Objective:** to make data accessible in a transparent manner
- **Technologies:** Open data, REST paradigms (HTTP based), MQTT (public-subscribe paradigm), cloud services
- **Requirements:** independence from underlying technologies, low complexity, ease of integration into applications, security, reliability

Process data

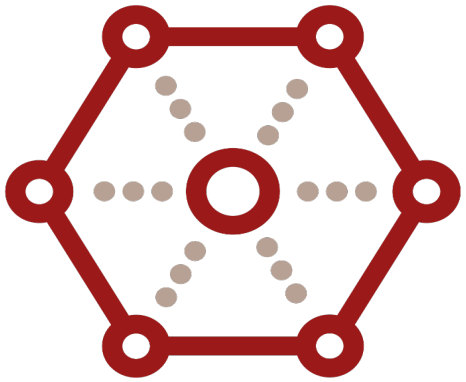
- **Objective:** Use the collected data to extract useful information, profiling, classification, time series prediction, automatic anomaly detection, ...
- **Technologies:** Data Analytics, Machine Learning, Data Visualization and Analysis Tools
- **Requirements:** ability to handle large data quays, ease of use of tools, ease of interpretation of results, reliability of results

Exploit data

- **Objective:** Use extracted information to build value-added services, process automation, predictive maintenance, context-aware systems self-configuration, ...
- **Technologies:** new business models (no longer based on data ownership but on sharing)
- **Requirements:** ability to integrate new information sources, security, privacy, ``functional ergonomics'' (acceptability and trust by the user)

DATA ACCESS TECHNOLOGIES

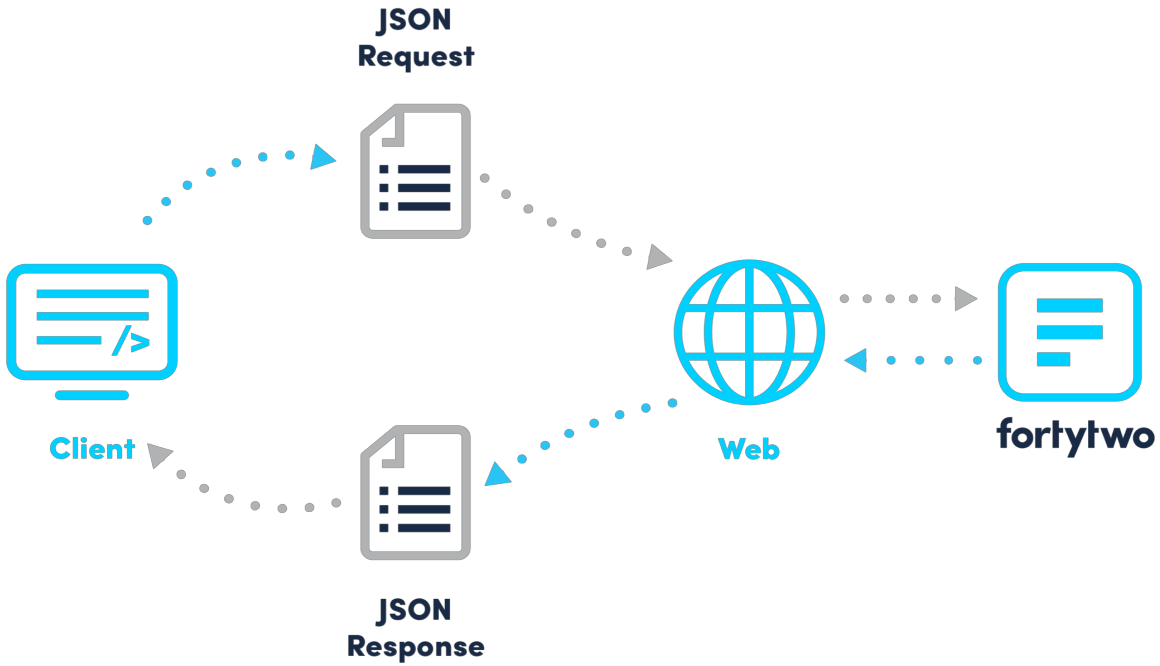
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE



RESTFULL PARADIGM

ReSTfull paradigm

- **REST**: Representational State Transfer
- Widely used; based on HTTP
- Lighter version: CoAP (Constrained Application Protocol)



RESTful

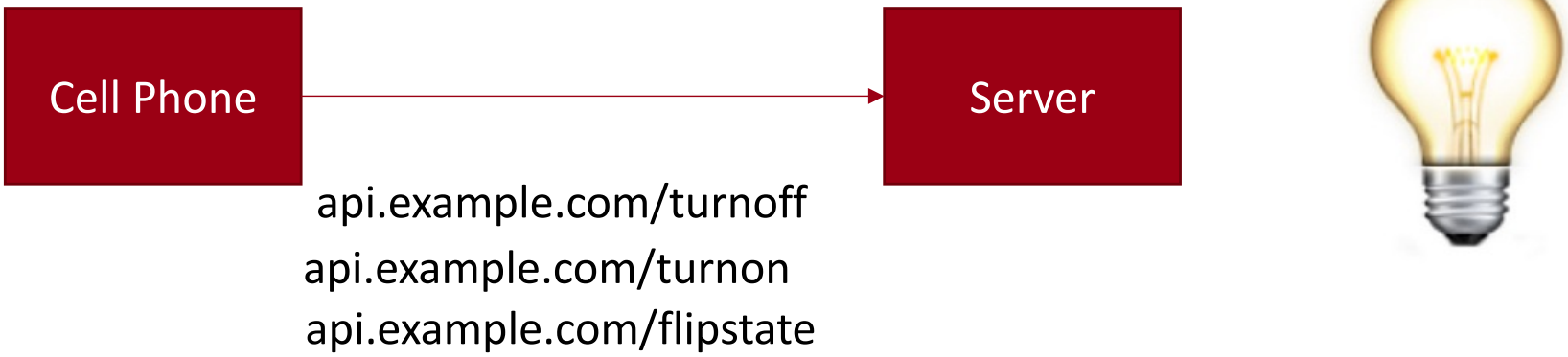
- A “request” is sent to a server via URL
  - http://api.example.com/resources/user/1036721?name=something

Path **or** variables

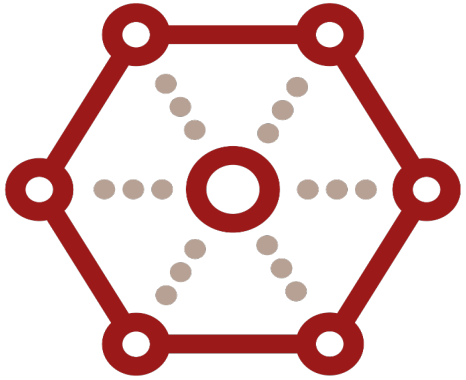
variables

- Response is usually text in HTML, XML, or JSON
- Great if you’re asking for something
  - What’s about “push”?
  - Used when server wants to tell device to do something

HTTP: RESTful Hypothetical Light Switch



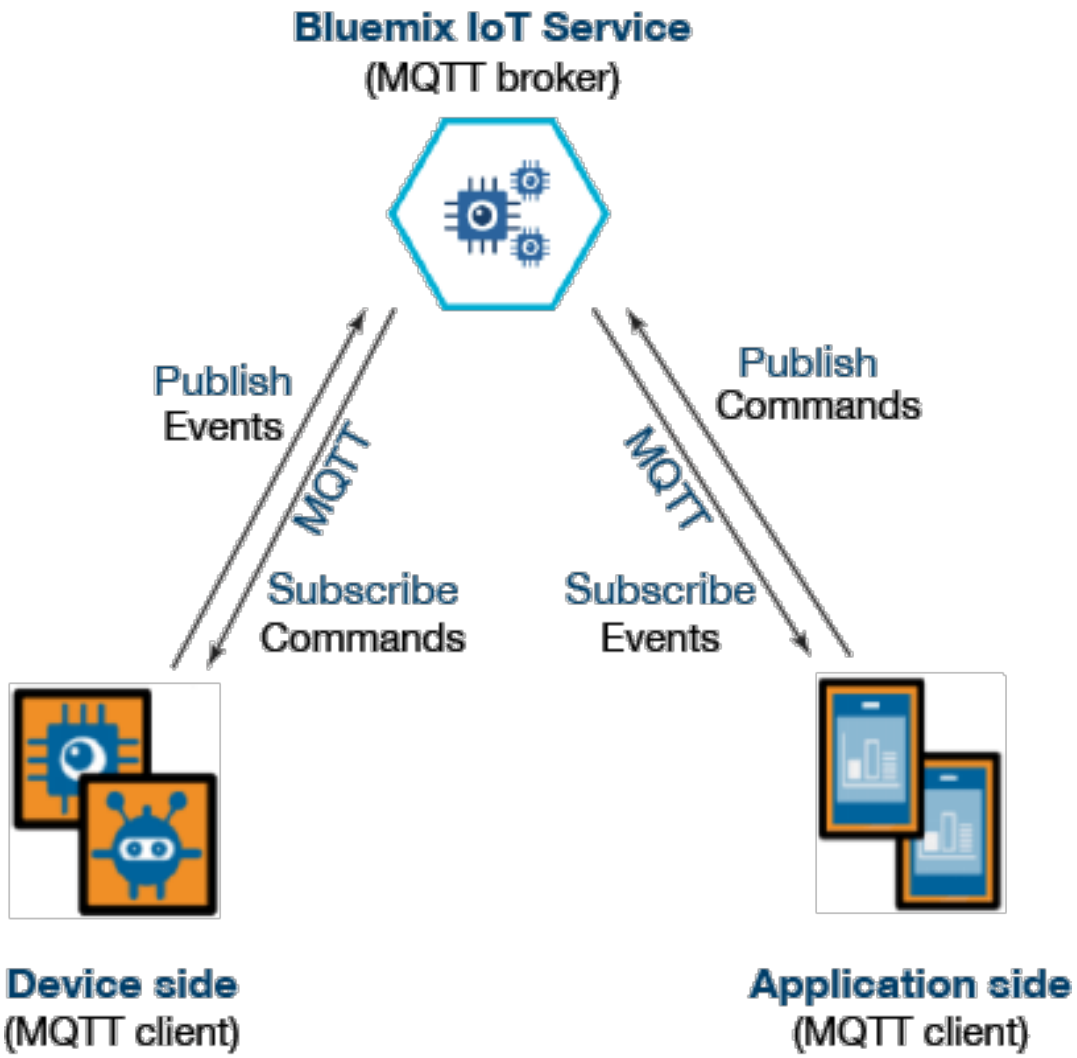
DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE



PUBLISH-SUBSCRIBE

Pub/sub approach

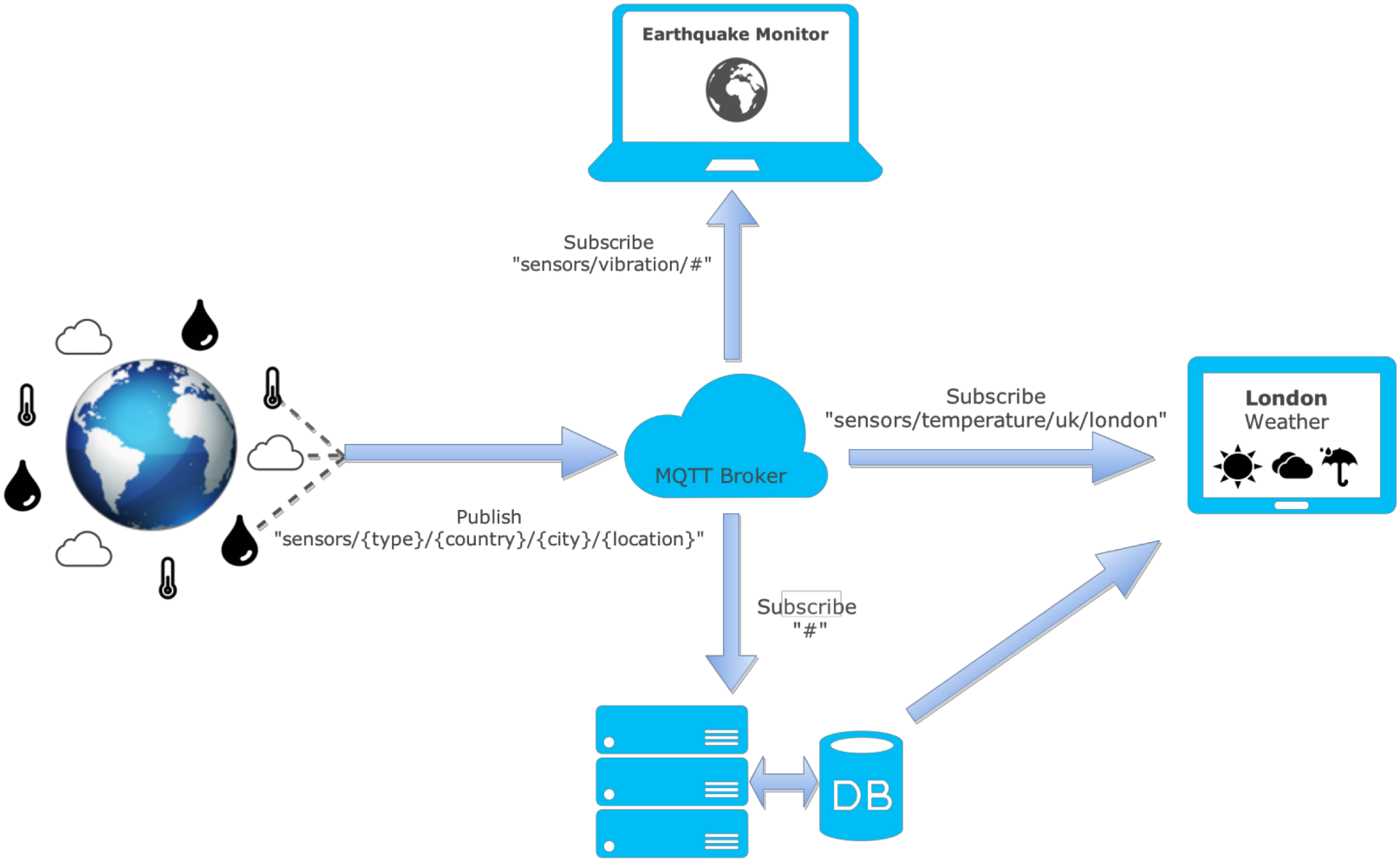
- Based on a publish-subscribe structure:
  - A **Publisher** sends messages according to Topics, to specified Brokers.
  - A **Broker** acts as a switchboard, accepting messages from publishers on specified topics, and sending them to subscribers to those Topics
  - A **Subscriber** receives messages from connected Brokers and specified Topics



Publish/subscribe

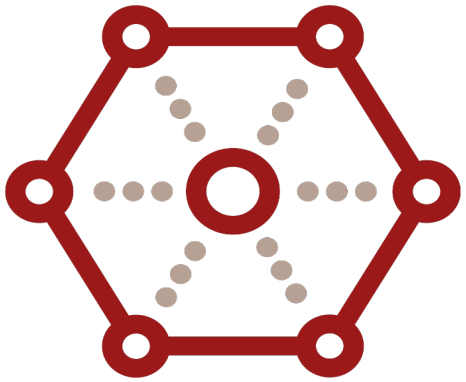
- **Multiple clients connect to a broker** and **subscribe** to topics that they are interested in
- **Clients** connect to the broker and **publish** messages to topics
  - Topics are treated as a hierarchy, using a slash (/) as a separator.
- Example: multiple computers may all publish their hard drive temperature information on the following topic, with their own computer and hard drive name being replaced as appropriate:
  - sensors/COMPUTER_NAME/temperature/HARDDRIVE_NAME
- Clients can receive messages by creating subscriptions
  - A subscription may be to an explicit topic, in which case only messages to that topic will be received, or it may include wildcards.
  - Two wildcards are available, + or #

Example of pub/sub architecture



Source: https://zoetrope.io/tech-blog/brief-practical-introduction-mqtt-protocol-and-its-application-iot

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE



MQTT

MQTT

- MQTT: Message Queuing Telemetry Transport
- Invented by Andy Stanford-Clark (IBM) and Arlen Nipper (Arcom, now Cirrus Link) back in 1999
  - their use case was to create a protocol for minimal battery loss and minimal bandwidth connecting oil pipelines over satellite connection
- They specified the following goals, which the future protocol should have:
  - Simple to implement
  - Provide a Quality of Service Data Delivery
  - Lightweight and Bandwidth Efficient
  - Data Agnostic
  - Continuous Session Awareness

Main characteristics

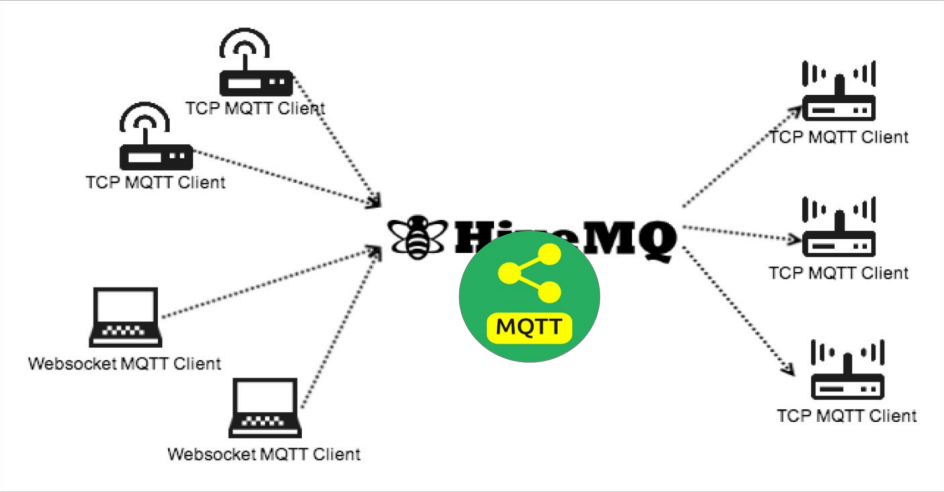
- A lightweight publish-subscribe protocol that can run on embedded devices and mobile platforms → http://mqtt.org/
- Low power usage
- Binary compressed headers
- Maximum message size of 256MB
  - not really designed for sending large amounts of data
  - better at a high volume of low size messages
- Documentation sources:
  - The MQTT community wiki:
    - https://github.com/mqtt/mqtt.github.io/wiki
  - A very good tutorial:
    - http://www.hivemq.com/mqtt-essentials/

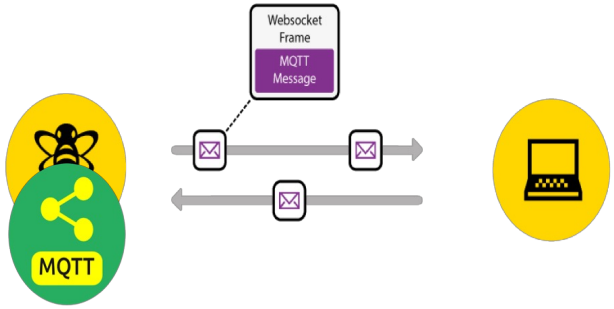
Some details about versions

- MQTT 3.1.1 is the current version of the protocol.
  - Standard document here:
    - http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
  - October 29th 2014: MQTT was officially approved as OASIS Standard.
    - https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt
- MQTT v5.0 is the successor of MQTT 3.1.1
  - Current status: Committee Specification 02 (15 May 2018)
    - http://docs.oasis-open.org/mqtt/mqtt/v5.0/cs02/mqtt-v5.0-cs02.html
  - Not backward compatible; too many new things are introduced so existing implementations have to be revisited, for example:
    - Enhancements for scalability and large scale systems in respect to setups with 1000s and millions of devices.
    - Improved error reporting (Reason Code & Reason String)
    - Performance improvements and improved support for small clients
  - https://www.youtube.com/watch?time_continue=3&v=YIpesv_bJgU

MQTT works on top of…

- Mainly of TCP
  - There is also the closely related MQTT for Sensor Networks (MQTT-SN) where TCP is replaced by UDP because TCP stack is too complex for WSN
- websockets can be used, too!
  - Websockets allows you to receive MQTT data directly into a web browser.





- Both, TCP & websockets can work on top of "Transport Layer Security (TLS)"

FEATUREs

- Small code footprint
- Ideal if processor or memory resources are limited
- Ideal if bandwidth is low or network is unreliable
- **Publish/subscribe message exchange pattern**
- Works on top of TCP/IP
- **Quality of service***: at most once*, *at least once*, *exactly once*
- Client libraries for Android, Arduino, C, C++, C#, Java, JavaScript, .NET
- Security: authentication using user name and password, encryption using SSL/TLS
- Persistence: MQTT has support for persistent messages stored on the broke
- MQTT-SN (protocol for sensor network) works on non-TCP/IP networks (e.g. Zigbee)
- MQTT over websocket possible (browser as MQTT client)
- Request/response message exchange pattern as add-on

MQTT: Pub/Sub

- Clients connect to a “Broker”
- Clients subscribe to topics eg,
  - client.subscribe(‘toggleLight/1’)
  - client.subscribe(‘toggleLight/2’)
  - client.subscribe(‘toggleLight/3’)
- Clients can publish messages to topics:
  - client.publish(‘toggleLight/1’, ‘toggle’);
  - client.publish(‘toggleLight/2’, ‘toggle’);
- All clients receive all messages published to topics they subscribe to
- Messages can be anything
  - Text
  - Images
  - etc

The Broker

- Broker is running on DigitalOcean
  - https://www.digitalocean.com/
- Broker is Mosquitto
  - https://mosquitto.org/
- Test
  - *https://test.mosquitto.org*
  - mosquitto_sub -h test.mosquitto.org -t "#" -v

Channels in MQTT

- Channels/topics in MQTT work like file paths

- When subscribing to a channel we have to specify the whole path
- Or use a **wildcard**
  - **+**
  - **#**

Message types

- CONNECT - Client request to connect to Server (broker)
- PUBLISH - Publish message
- SUBSCRIBE - Client Subscribe request

QoS level

- At most once: no ACK expected from receiver
- At least once: the message will continue to be resent at regular intervals, until the sender receives an acknowledgement.
- Exactly once: msg delivered one and only one time to upper layer (four way handshake)

Summing up

- MQTT (pub-sub) easy to use, robust and scales well for one to many comm.
- Fast
  - 0.04 s for message delivery
- Light
- Efficient
- Widespread