The MQTT Protocol

The MQTT protocol implements a **publish-subscribe messaging** mechanism, involving three main actors

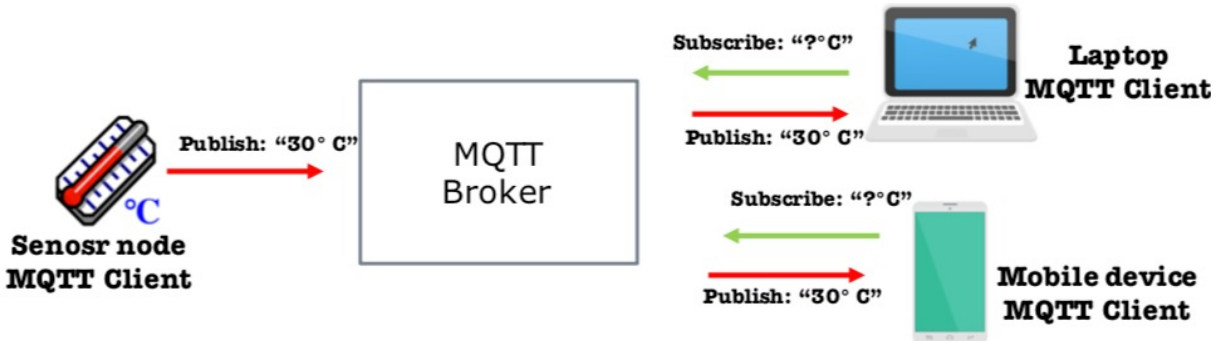
- **Publishers:** produce data and send them to a broker.
- **Subscribers:** subscribe to a topic of interest, and receive notifications when a new message for the topic is available.

Clients

- **Broker:** filter data based on topic and distribute them to subscribers.
  - Mosquitto https://mosquitto.org/
  - HiveMQ https://www.hivemq.com/public-mqtt-broker/

Publish/Subscribe paradigm

- Clients don't know each other
- One-to-Many paradigm
- Every client publishes & subscribes
- PUSH information paradigm compared to PULL's one in COAP



COnstrained Application Protocol (COAP)

- GOAL: to enable web-based services in constrained wireless networks
  - 8 bit micro-controllers
  - limited memory
  - low-power networks
- Problem: WEB solution are hardly applicable
- Solution: re-design web-based services for constrained networks -> COAP

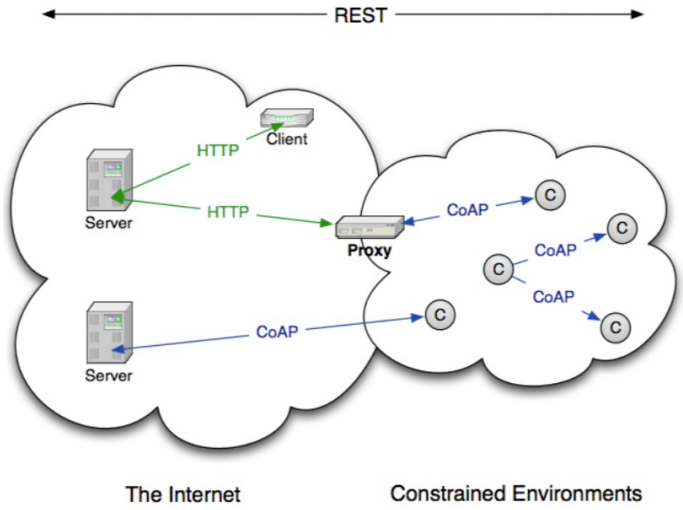
- Resources in the Web are:
  - managed by servers
  - identified by URIs (Uniform Resource Identifier)
  - accessed asynchronously by clients through request/response paradigms
- In a word, Representational State Transfer (REST)

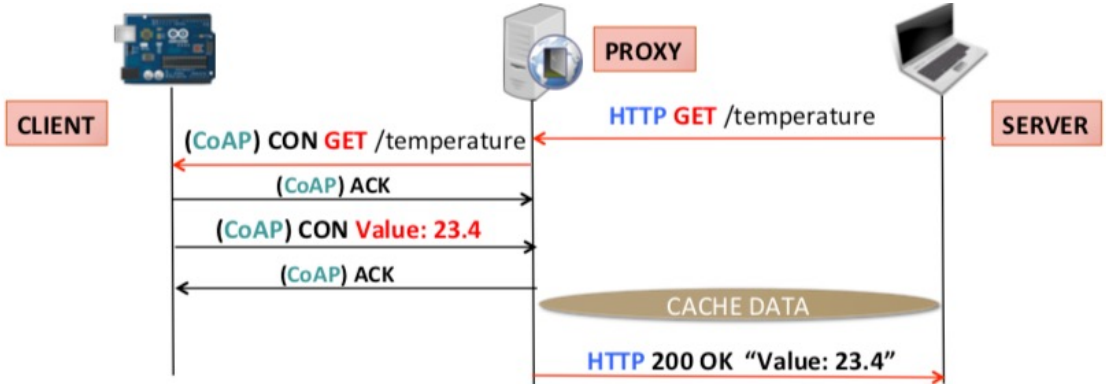
- Differently from MQTT, CoAP implements a **request-response interaction** model (similar to the HTTP protocol).

The CoAP Architecture

Proxying and caching

- CoAP only supports a limited subset of HTTP functionality,
- However, cross-protocol proxy mechanisms can guarantee seamless **HTTP-CoAP interactions** (beside providing data caching).





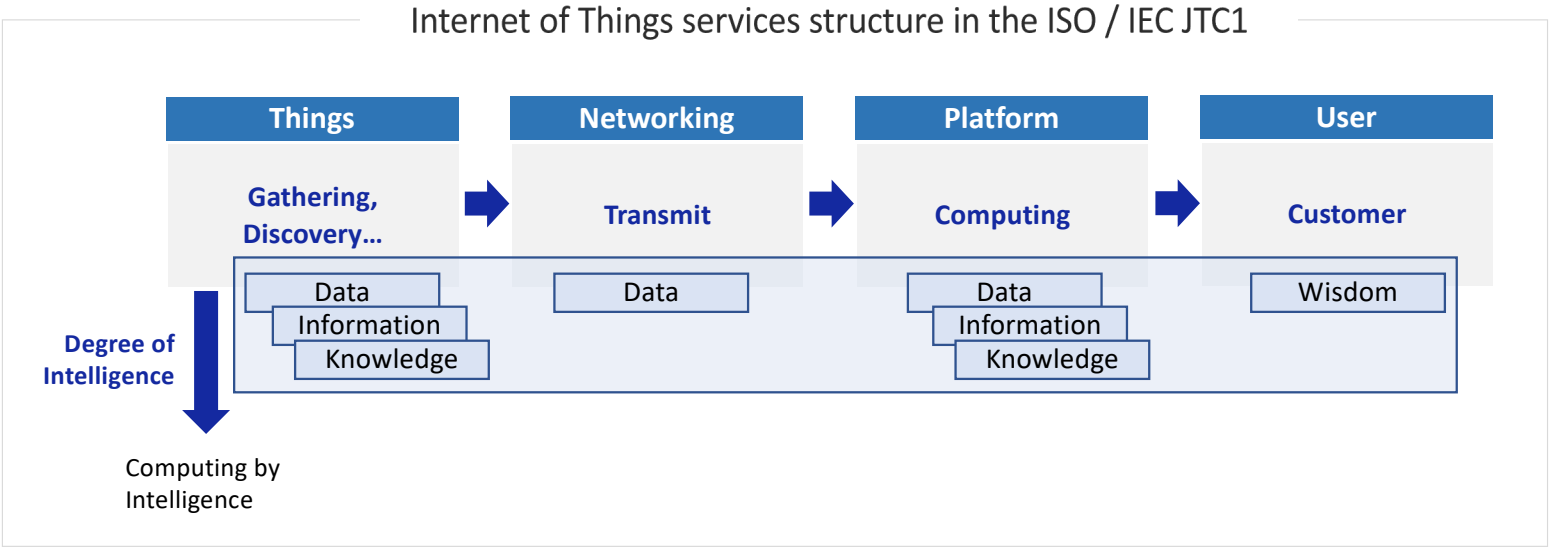
IoT Platform

...in the cloud

Basic Structure of the IoT Platform

IoT platform requirements

- By providing common functions required by things and services in the IoT service structure, various users must be capable of producing and managing services. Convenience should be provided for customers (developer, service user) who will use the service.

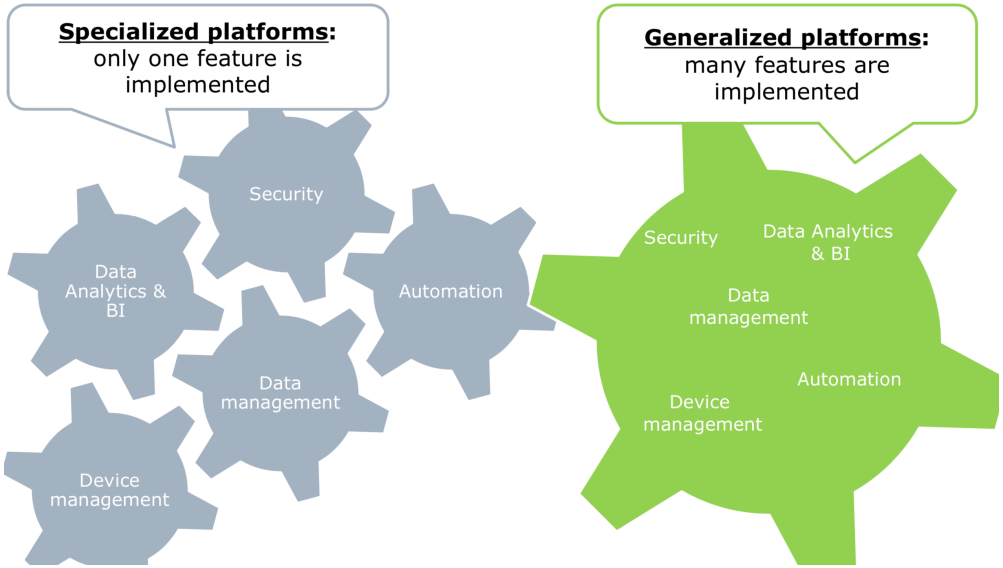


Desired features of IoT platforms

Device management	Data management	Data analysis / automation	Security
• Connect devices to the cloud • configure devices, • update firmware, • monitor devices...	• Store and retrieve data, • manage events, • visualize and share data	• Statistical analysis, • data mining, • machine learning, etc...	• Confidentiality • Integrity (data and system) • Availability • Authenticity • Accountability

Examples of IoT Platforms

- Old Giants
  - IBM: Bluemix IoT Cloud, Node-RED
  - Microsoft: Azure IoT Suite
  - Amazon: AWS IoT
  - Google: Cloud IoT + Brillo
  - Intel: IoT platform
- Startups
  - Xively, Thingspeak, Freeboard, ThingWorx, The Things Networks, …



Specialized vs Generalized platforms

Specialized

- May be difficult to link to a particular feature
  - E.g. “Device Management” can be very specific (depending on the device)
- Security (device&data) is growing importance

Generalized

- Two main approaches
  - Native IoT platforms: former specialized platforms that evolved with the addition of new features (e.g., data management)
  - Cloud-based IoT platforms: traditional cloud-based solutions that evolved with specialized features (e.g., device management) for the IoT
- Strengths:
  - Excellent data management and interface with corporate IT solutions

How to choose?

- Specialized: Connectivity management, Device configuration, OtA firmware update
- Generalized: Data mining & machine learning, Data fusion, Event processing, Rules engine

The choice is application-dependent!