Live-Objects

Alessandro Danielli
alessandro.danielli@edu.esiee.fr

Index

1. **Overview**
   - 1.1 What is Live-Objects
   - 1.2 Inside Live-Objects
   - 1.3 Concepts
2. **Architecture**
   - 2.1 Connectivity Layer
   - 2.2 Bus Layer
   - 2.3 Device Management
3. **Functionalities**
   - 3.1 Simple Event Processing
   - 3.2 State Processing
   - 3.3 Activity Processing
4. **MQTT Example**

Ref: [16]

WHAT IS…

Live Objects is **software as a service** (SaaS) who provides a set of tools for Machine To Machine (M2M) and Internet of things (IoT).

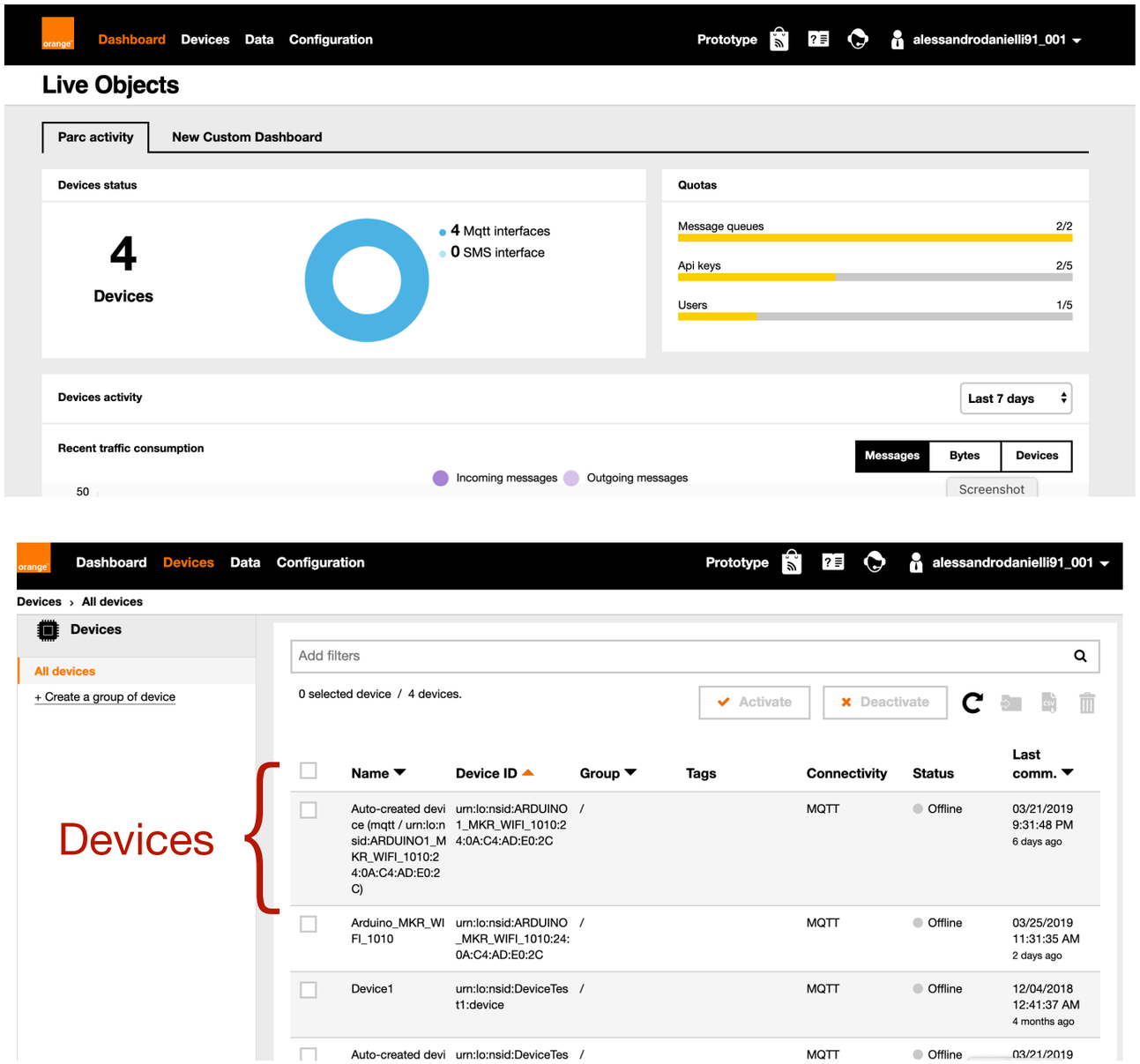
It is a platform suitable for IoT/M2M application offering

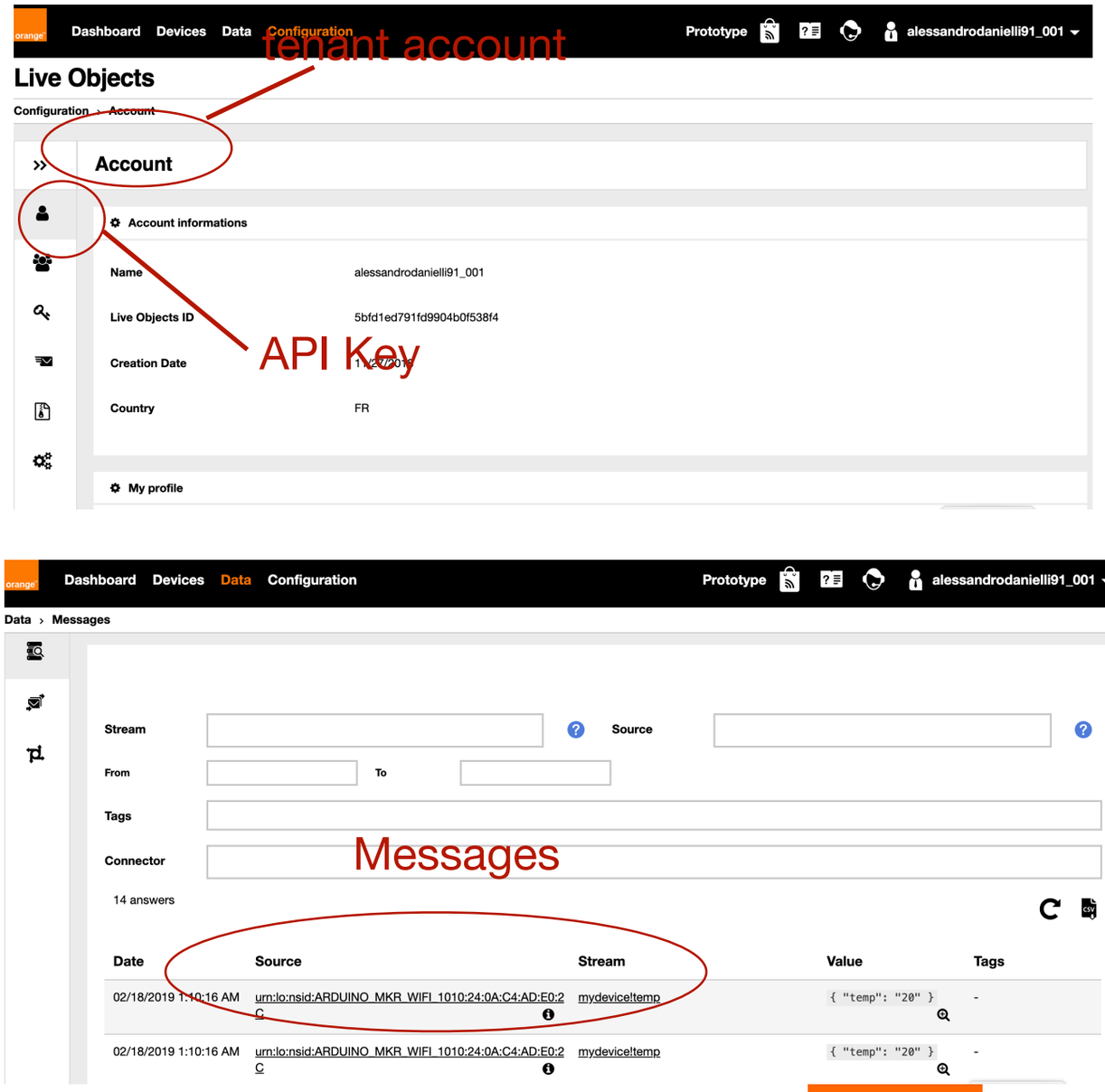
- **Connectivity Interfaces** (public and private) to collect data, send command or notification from/to IoT/M2M devices,
- **Device Management** (supervision, configuration, resources, firmware, etc.),
- **Message Routing** between devices and business applications,
- **Data Management:** Data Storage with Advanced Search features.

Ref: [https://liveobjects.orange-business.com/ (10/2018)]

Overview

INSIDE LIVE-OBJECTS





Ref: [https://liveobjects.orange-business.com/ (10/2018)]

CONCEPTS

A **tenant account** is the isolated space on Live Objects dedicated to a specific customer: every interaction between Live Objects and an external actor (user, device, client application, etc.) or registered entities (user accounts, API keys, etc.) is associated with a tenant account. The tenant account has:

- tenant ID: unique identifier
- name

A Live Objects **API key** is a secret that can be used by a device/app/user to authenticate when accessing to Live Objects on the MQTT or HTTP/REST interfaces.

An API key belongs to a tenant account and an API key can have zero, one or many roles. These roles allows to restrict the operations that could be performed with the key.

A **Role** can be attributed to an API key or user account. It defines the privileges of this user or API key on Live Objects.

Every interaction between Live Objects and devices and applications is modeled as one or many "**messages**". Messages can be written in two ways, as normal text (less efficient) or in JSON format (suitable for IoT application).

A **device** is represented by an unique identifier. This identifier must respect the following format:

urn:lo:nsid:{ns}:{id}

ns: "namespace" used to avoid conflicts between various families of identifier
id: device identifier

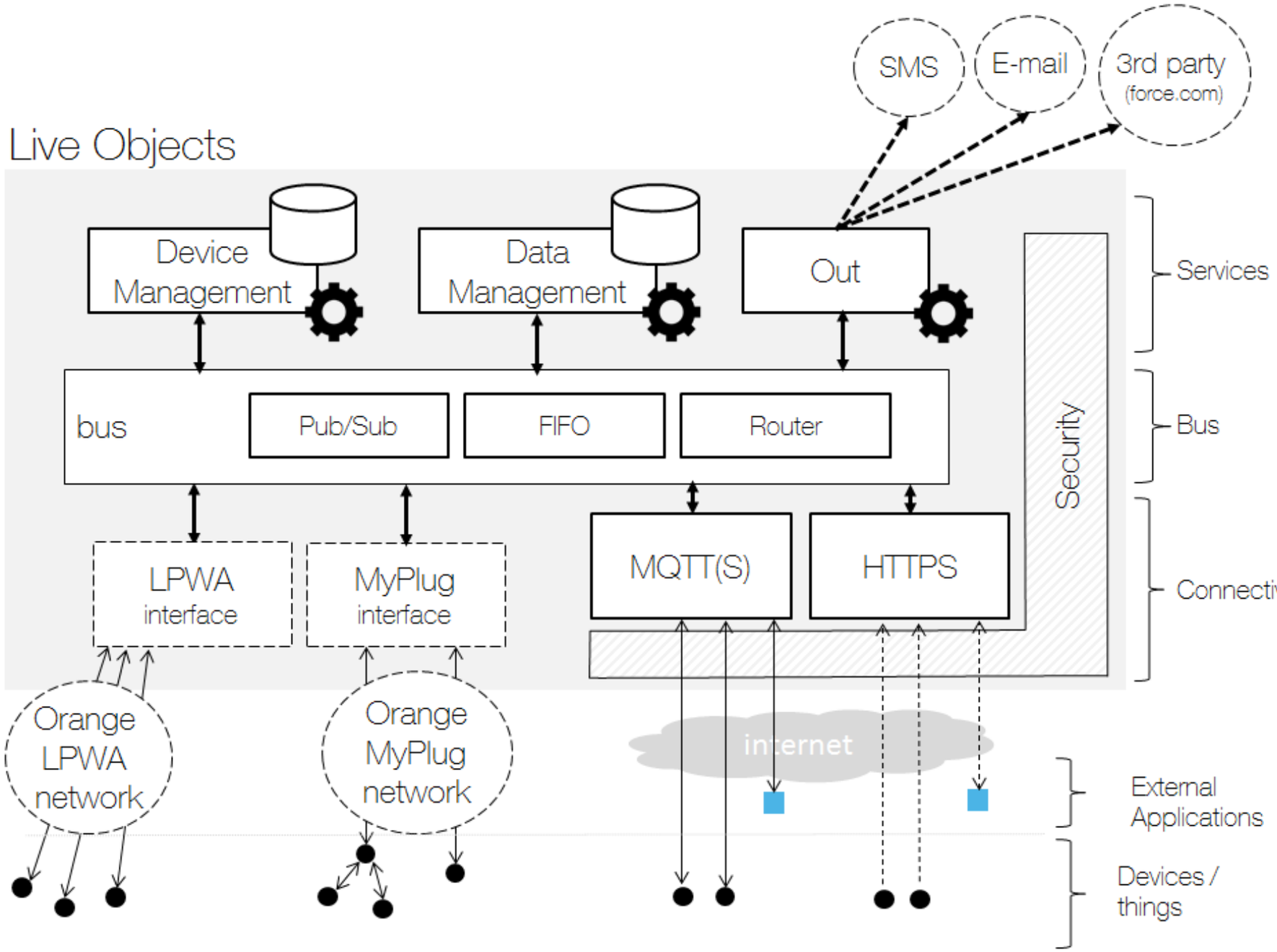
Ref: [https://liveobjects.orange-business.com/ (10/2018)]

Overview

Role Name	Technical value	Admin profile	User profile	Priviledges
API key	**API_KEY_R**	X	X	Read parameters and status of an API key.
API key	**API_KEY_W**	X	X	Create, modify, disable an API key.
User	**USER_R**	X	X	Read parameters and status of a user.
User	**USER_W**	X		Create, modify, disable a user.
Settings	**SETTINGS_R**	X	X	Read the tenant account custom settings.
Settings	**SETTINGS_W**	X	X	Create, modify tenant account custom settings.
Device	**DEVICE_R**	X	X	Read parameters and status of a Device (aka Asset).
Device	**DEVICE_W**	X		Create, modify, disable a Device (aka Asset), send command, modify config, update ressource of a Device.

Ref: [https://liveobjects.orange-business.com/ (10/2018)]

Architecture



Ref: [https://liveobjects.orange-business.com/ (10/2018)]

Architecture - Connectivity Layer

CONNECTIVITY LAYER: set of interfaces allowing to connect any programmable devices.

Interface

An interface is a platform access. A device can have no, one or several interfaces, which represent different device connectivities used to communicate with Live Objects.

Public Interfaces: Set of standard and unified public interfaces reachable from internet, they allow to connect any programmable devices

- MQTT(S) interface to communicate in a bidirectional way.
- HTTPS interface to communicate in a unidirectional way.

Private Interfaces: They provide interfaces with a selection of devices (MyPlug) or specific network (LPWAN). They handles communications from specific families of devices with defined protocol (over IP), and translate them as standardised messages available on Live Objects message bus.

- LPWAN interface
- MyPlug interface

(MyPlug: https://www.orange.com/sirius/hello/2013/en/2012-innovations/my-life-at-home/my-plug.html)

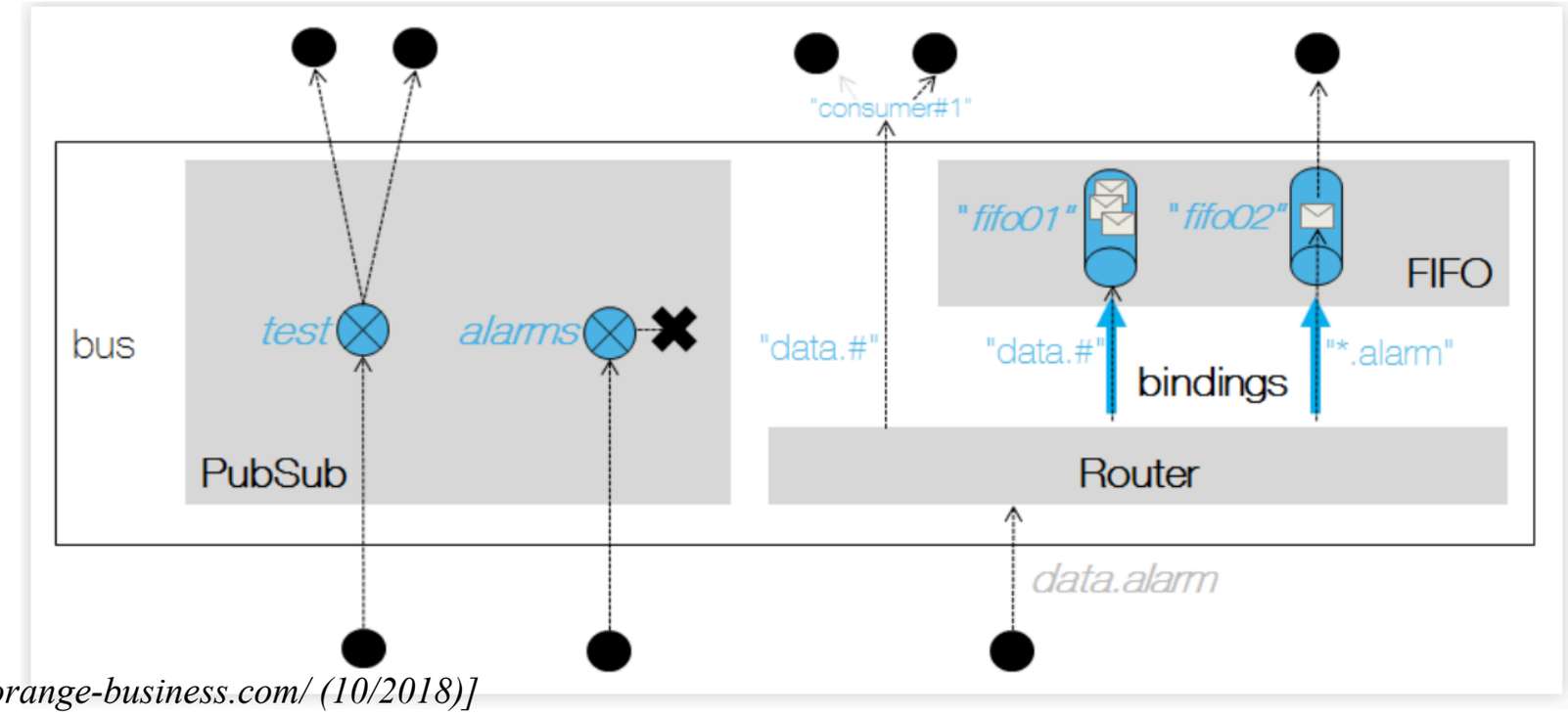
Ref: [https://liveobjects.orange-business.com/ (10/2018)]

BUS LAYER

Live Objects connectivity interfaces are connected to a message bus that could route message to external business application or internal micro-services (device management, store and search services).

Three modes:

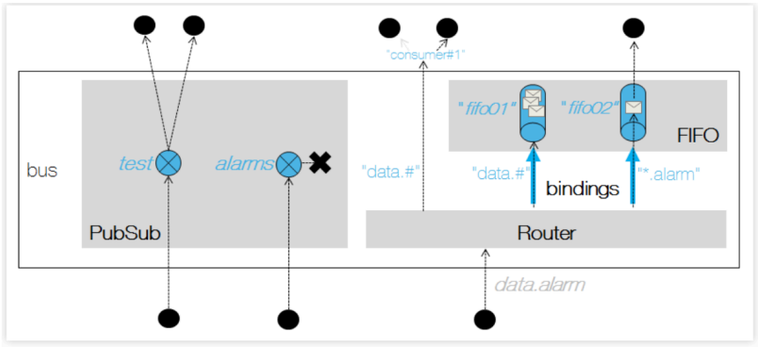
- **Router** : adapted to situations where publishers don't know the destination of the messages.
- **PubSub** : a good fit for real-time transient exchanges. Message are broadcast to all currently available subscribers or dropped.
- **FIFO** : the solution to prevent from message loss in the case of consumer unavailability. Messages are stored in a queue on disk until consumed and acknowledged.



Ref: [https://liveobjects.orange-business.com/ (10/2018)]

Router Mode

- (At the bottom) A client publishes into the Router a message with routing key *data.alarm.*
- (On the left) Two clients are subscribed on router with routing key filter *data.#* and consumer identifier *consumer#1*. As the routing key filter matches the routing key of the published message, the message is delivered to those clients. If clients are subscribed with the same consumer id, the message is *load balanced* : only one of the two consumers receives the message.
- (At the center) A binding with routing key filter *data.#* is declared from the Router to the FIFO queue "fifo01": this routing key filter matches the routing key of the published message so the message is delivered to this FIFO queue as if it was published in FIFO mode to topic "fifo01".
- (On the right) A binding with routing key filter **.alarm* is declared from the Router to the FIFO queue "fifo02": this routing key filter matches the message routing key, so the message is delivered to the FIFO. As a subscriber is currently subscribed to the FIFO queue, it immediately receives the message, but the message is also stored on disk into the queue until acknowledged.



Ref: [https://liveobjects.orange-business.com/ (10/2018)]

PubSub mode

Communications in PubSub mode is based on the usage of topics. A **topic** is uniquely identified by a string with the following format: "*<topic type>/<topic name>*". Where *<topic type>* can be "*pubsub*" or "*fifo*", and *<topic name>* is an arbitrary string.

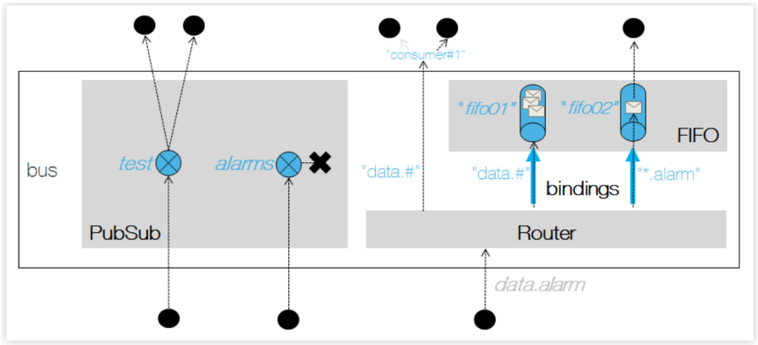
```
"pubsub/alldevices" or "fifo/alerts"
```

In PubSub mode clients can publish or subscribe to one or many topics. When a client publishes a message on a specific PubSub topic, the message is broadcast in real-time to all currently subscribed clients. The message is not persisted by Live Objects messaging layer: if no consumers have subscribed, the message is simply dropped and lost forever.

- (On the left) A client publishes on PubSub topic 'test' while two consumers are subscribed, the message is then duplicated and delivered to the two consumers.

- (On the right) A client publishes on PubSub topic 'alarm' while no consumers are subscribed: the message is dropped.

The PubSub mode is a good fit for the following patterns:
- broadcasting non-critical events to groups of consumers,
- one-to-one real-time dialogs



Ref: [https://liveobjects.orange-business.com/ (10/2018)]

FIFO mode

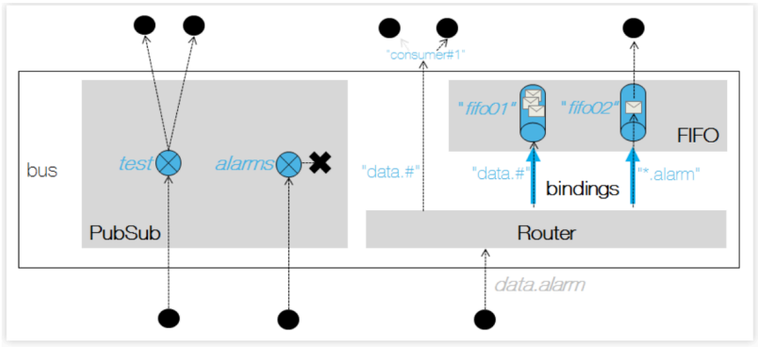
FIFO mode communication is also based on the usage of topics. There are no conflict between the naming of PubSub topics and FIFO topics (the PubSub topic "test" is different from FIFO topic “test”).

Messages published on a FIFO topic are persisted until a subscriber is available and acknowledges the handling of the message. If multiple subscribers consume from the same FIFO topic, messages are load balanced between them. Publication to and consumption from a FIFO topic use acknowledgement, ensuring no message loss.

- (On the left) A client publishes in FIFO topic/queue "fifo01" while no consumer is subscribed. The message is stored into the queue, on disk. When later a consumers subscribes to the FIFO topic/queue, the message will be delivered. The message will only disappear from disk once a subscriber acknowledges the reception of the message.

- (On the right) A client publishes on FIFO topic/queue "fifo02" while a consumer is subscribed: the message is stored on disk and immediately delivered to the consumer. The message will only disappear from disk once a subscriber acknowledges the reception of the message. When a consumer that received the message but didn’t acknowledged the message unsubscribes from the topic/queue, the message is put back into the "fifo02" queue and will be delivered to the next available consumer.

FIFO are size-limited as the number of FIFOs and it depends on your offer.



Ref: [https://liveobjects.orange-business.com/ (10/2018)]

Device management

- Register a device
- Configuration
- Commands

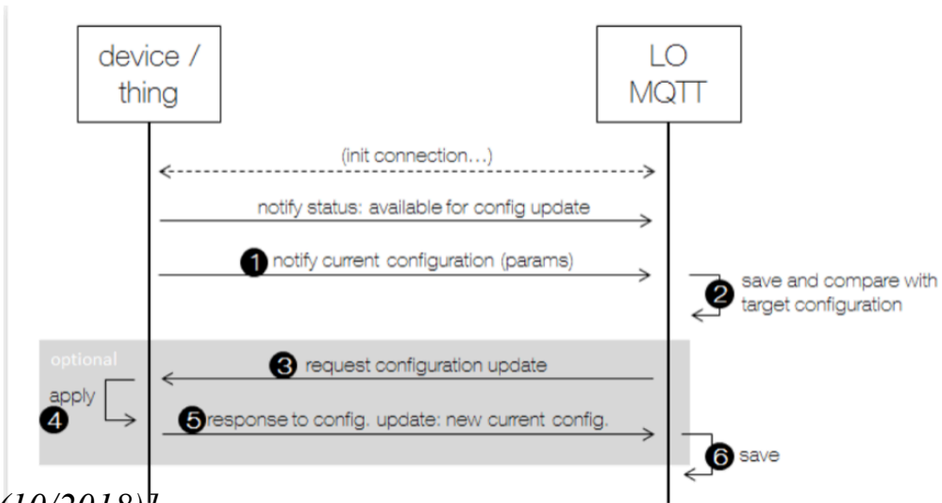
Register a device: In order to be able to management the devie through Live Objects you need to register it.
For MQTT devices the registration and the relative interface could be done manually by the user or automatically by Live Objects at the first connection.
For LPWA devices it is user's responsibility to associate an interface to the corresponding device.

```
{
    "id": "urn:lo:nsid:lora:0101010210101010",
    "tags": ["Lyon", "Test"],
    "name": "myLoraSensor",
    "description": "device with LoRa interface",
    "properties" : {
        "manufacturer": "Orange",
        "model": "LoraSensor"
    },
    "interfaces": [
        {
            "connector": "lora",
            "enabled": true,
            "definition": {
                "devEUI": "0101010210101010",
                "profile": "Generic_classA_RX2SF9",
                "activationType": "OTAA",
                "appEUI": "9879876546543211",
                "appKey": "11223344556677889988776655443322",
                "connectivityOptions" : {
                    "ackUl" : true,
                    "tdoa" : false
                }
            }
        }
    ],
    "group": {
```

Ref: [https://liveobjects.orange-business.com/ (10/2018)]

Device configuration:

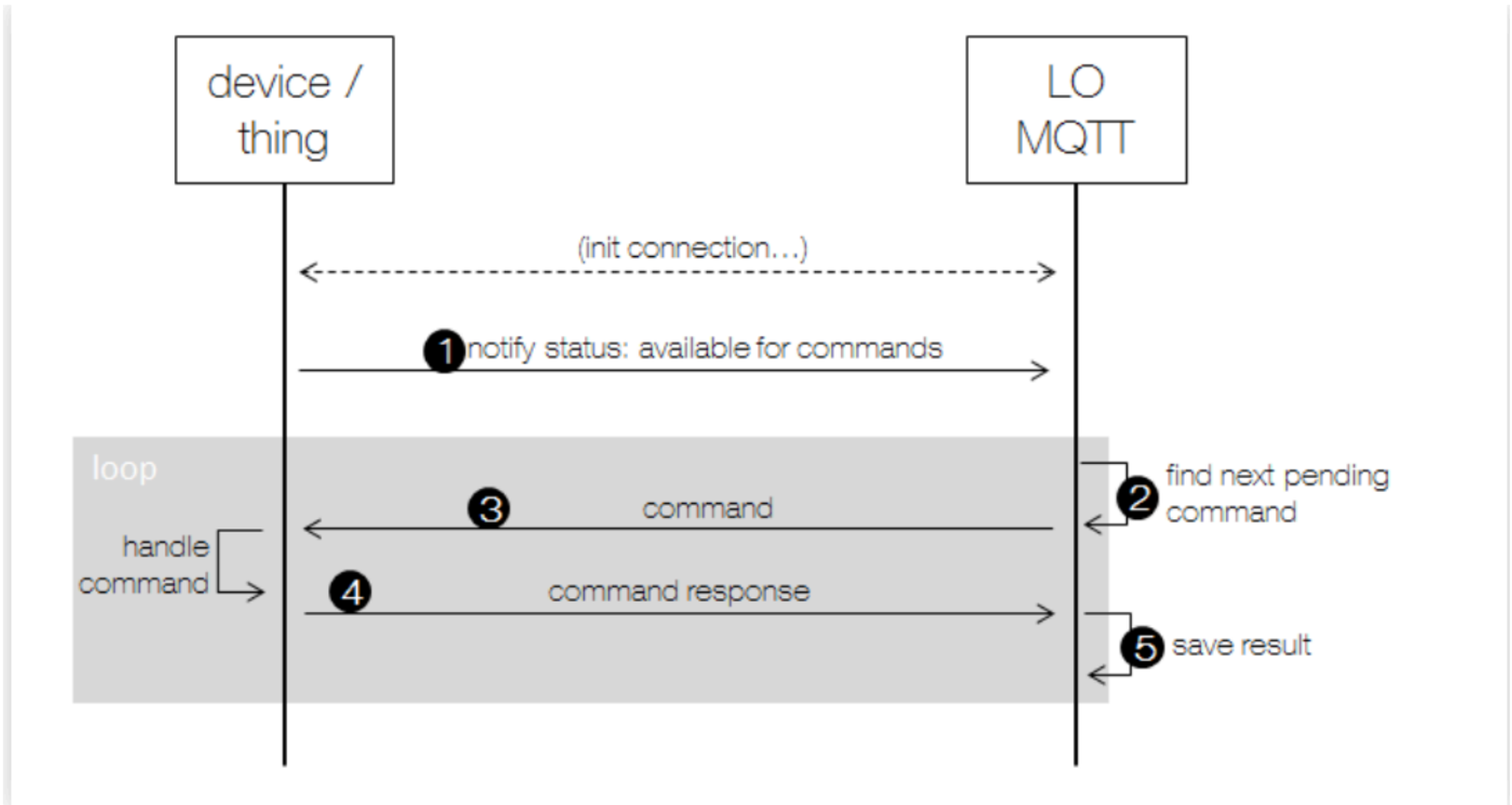
- (before) :
  - device initiates MQTT connection with Live Objects,
  - device subscribes in MQTT to a private topic, where it will receive later the configuration update requests,
- **step 0** : device notifies Live Objetcs that it is connected and available for configuration updates on a specific topic,
- **step 1** : device notifies Live Objetcs of its current configuration,
- **step2** : Live Objects compares the current and target configuration for this device. If they differ:
  - **step 3** : Live Objects sends to the device, on the topic indicated at step 0, the list of parameters to update, with their target value,
  - **step 4** : device handles the request, and tries to apply the change(s),
  - **step 5** : device respond to the change request with the new configuration,
  - **step 6** : Live Objects saves the new configuration. Parameters that have been successfully updated now have the status "OK" and the others the status "FAILED".



Ref: [https://liveobjects.orange-business.com/ (10/2018)]

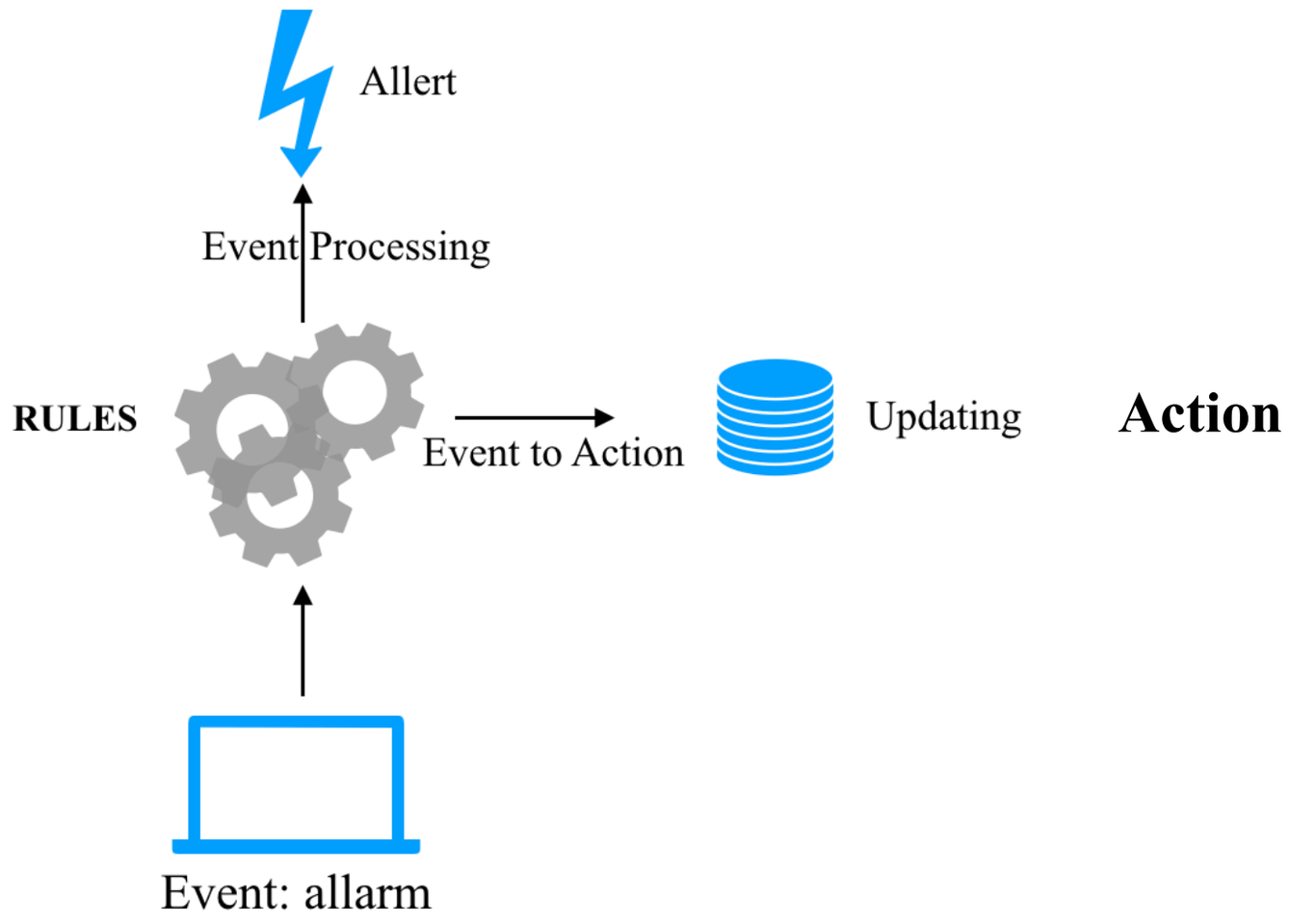
Commands:

You can register commands targeting a specific device: as soon as the device is available for commands, Live Objects will send them one by one, waiting for a response for each command from the device before sending the next one.



Ref: [https://liveobjects.orange-business.com/ (10/2018)]

Event: everything happens in my device that get my attention



Ref: [https://liveobjects.orange-business.com/ (10/2018)]

Functionalities

Event processing

3 independent features are available for event processing:

Simple Event Processing [SEP]

- Detecting *single event*
- *Stateless* connection

Example: raise an event when the device temperature is above 25°C.

State Processing [SP]

- Detecting changes in *device state*
- *Stateful* connection

Example: raise an event when the device temperature status changes from "cold" to "hot".

Activity Processing [AP]

- Detecting silent devices
  'ACTIV'
  'SILENT'
  'UNKNOWN'

Example: raise an event when the device is not connected or did not send data message since 1 day.

https://www.youtube.com/channel/UCqiOhlRlpjRvR3Bw0hMLciw

Ref: [https://liveobjects.orange-business.com/ (10/2018)]

Functionalities

Simple Event processing

It detects single events from the flow of data messages with the used of a stateless boolean detection function, called **matching rule**, and a frequency function, called **firing rule**.

Matching rule: rules applied to each data message in order to evaluate if a specific condition occurs. These rules are store in the matching context DataBase and they are written in JSON format. Once the rules has been applied the data flows passes through the firing rule.

Firing rule: for each matching event created by the matching rule function a firing rule is applied. It define when or how often the event must be generated. There are 3 possible choices: ONCE, SLEEP, ALWAYS.

If ONCE or SLEEP is applied the system generate a **firing guard** to prevents the generation of new firing events. With SLEEP mode is possible to specify after how much time the firing guard can be removed.

State processing

the detection function **state rules processing** is used to detect changes in device state. A state is computed by applying a state function to a data message. A notification is sent by Live Objects each time a state value change.

Ref: [https://liveobjects.orange-business.com/ (10/2018)]

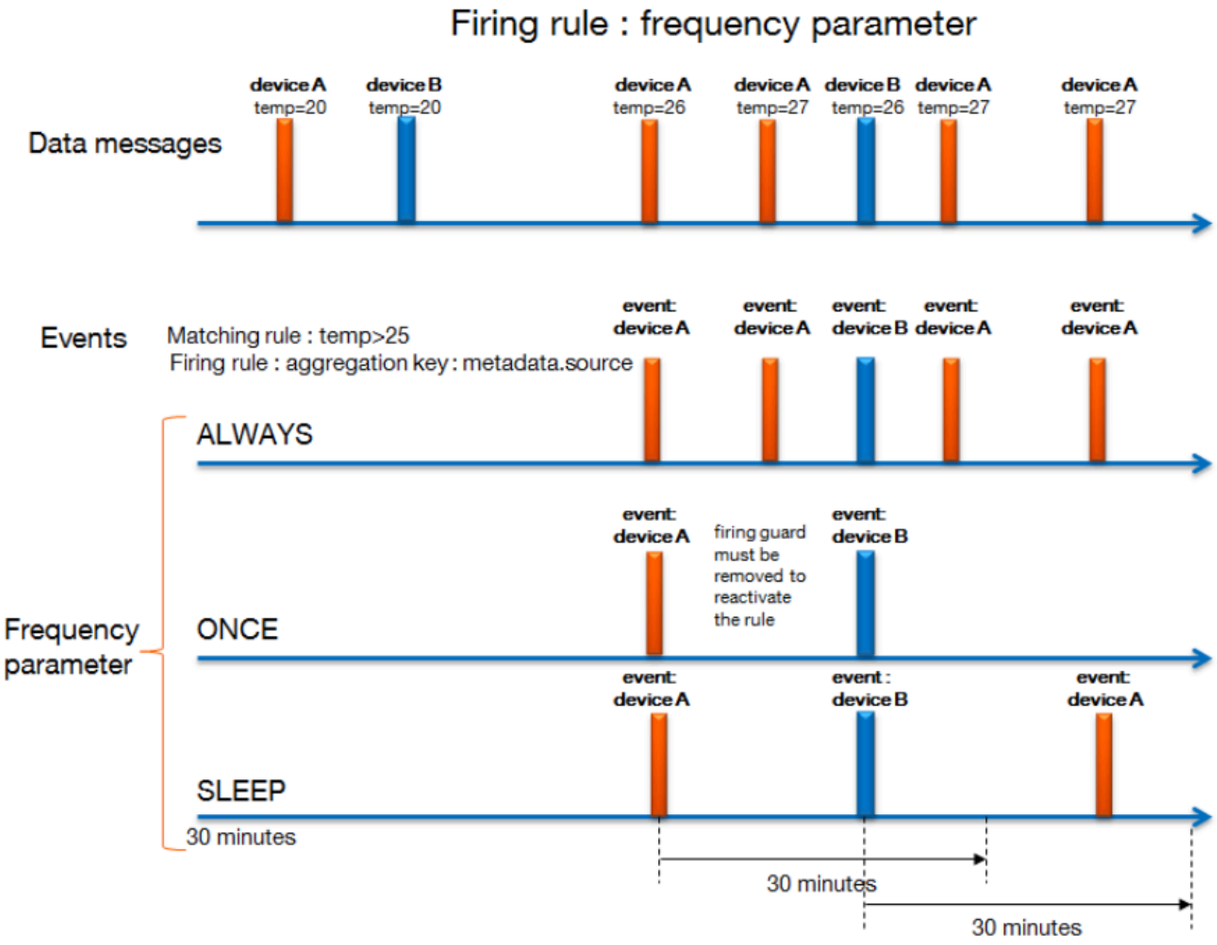
Functionalities

Pollutant	Threshold	
	INFO	ALERT
Nitrogen dioxyde NO2	hourly average: 200 µg/m³	400 µg/m³ over 3 consecutive hours or 200 µg/m³ if previous day pollution level=MEDIUM or HIGH
Particulate Matter PM10	hourly average: 50 µg/m³	hourly average: 80 µg/m³

Simple Event processing

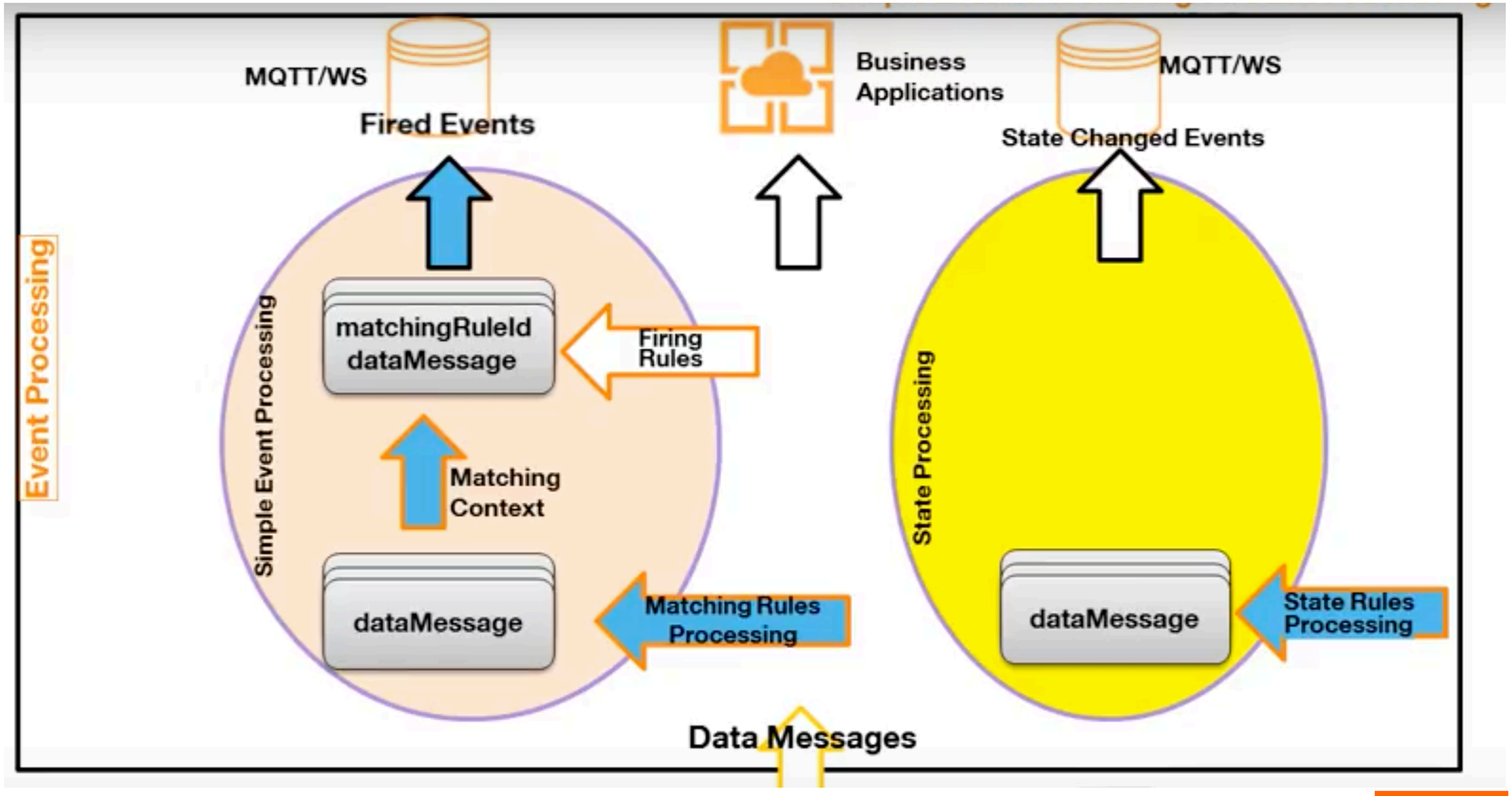
Pollutant	Daily pollution level (daily average)		
	LOW	MEDIUM	HIGH
Nitrogen dioxyde NO2	<200 µg/m³	between 200 and 400 µg/m³	>400 µg/m³
Particulate Matter PM10	<50 µg/m³	between 50 and 80 µg/m³	>80 µg/m³

State processing



Ref: [https://liveobjects.orange-business.com/ (10/2018)]

Functionalities



Ref: [https://liveobjects.orange-business.com/ (10/2018)]

MQTT Example

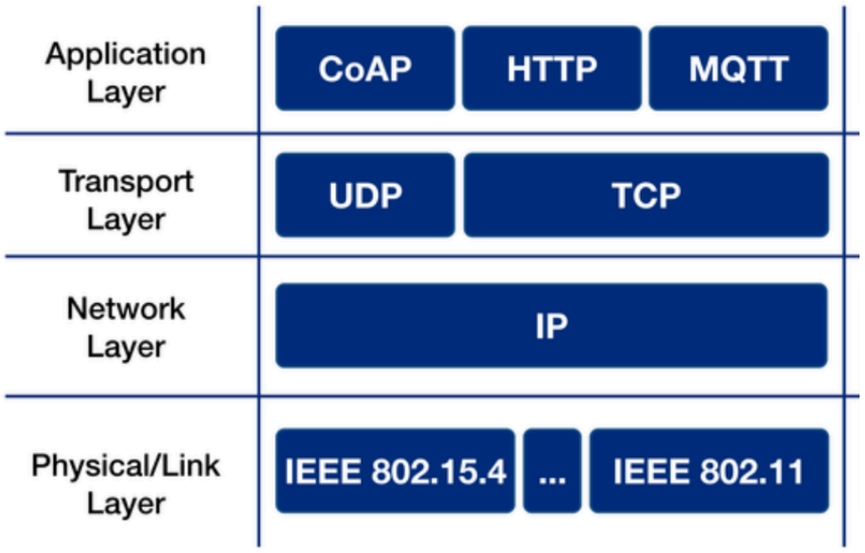
- **MQTT protocol**
- **Application Design**
- **Connection Parameters**
- **Send/Received Data**

MQTT Overview

Message Queue Telemetry Transport is a "lightweight" Client-Server publish/subscribe messaging transport protocol. Its characteristics make it ideal for use in Machine to Machine (M2M) communications and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is a constrained.

FEAUTERS:

- It is an application protocol in the ISO model
- publish / subscribe' paradigm
- It is based on TCP/IP or or over other network protocols that provide ordered, lossless, bi-directional connections.
- 3 QoS for message delivery
- Small transport overhead and protocol exchange to reduce network traffic



Protocol stack

Ref: [http://mqtt.org/ (10/2018)]

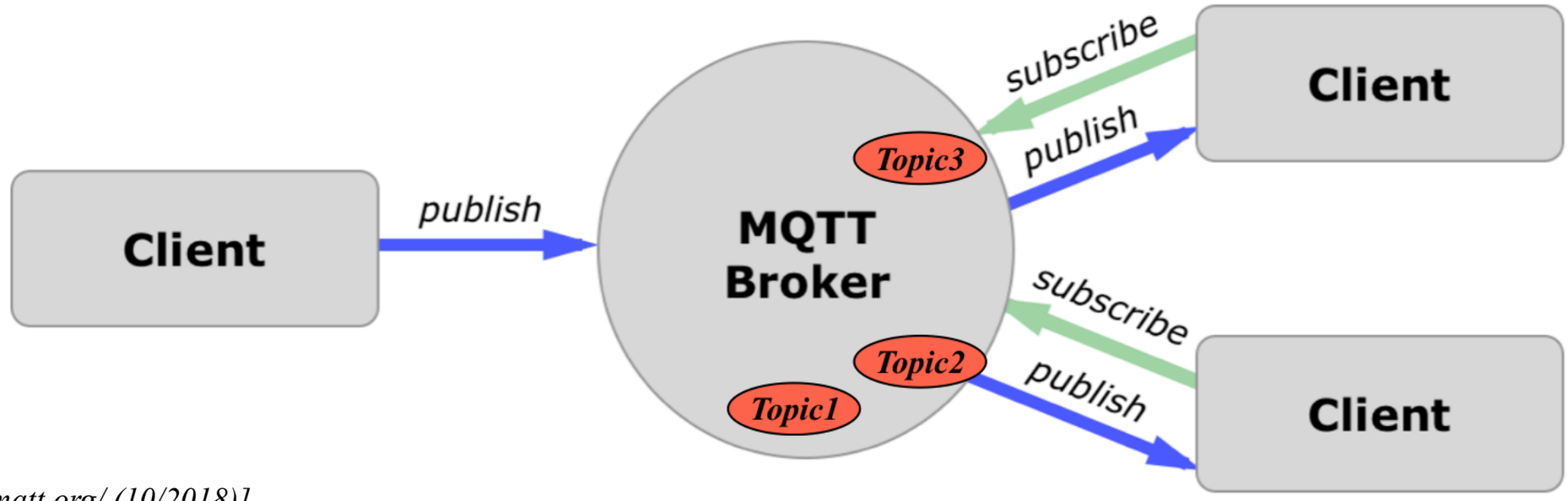
MQTT Overview

PUBLISH / SUBSCRIBE paradigm: it provides one-to-many messages distribution in the network.

A Client can: - Publish messages that other Clients might be interested in. (send data)
- Subscribe to topic that it is interested in receiving. (receive data)
- Unsubscribe to a topic
- Disconnect from the Server

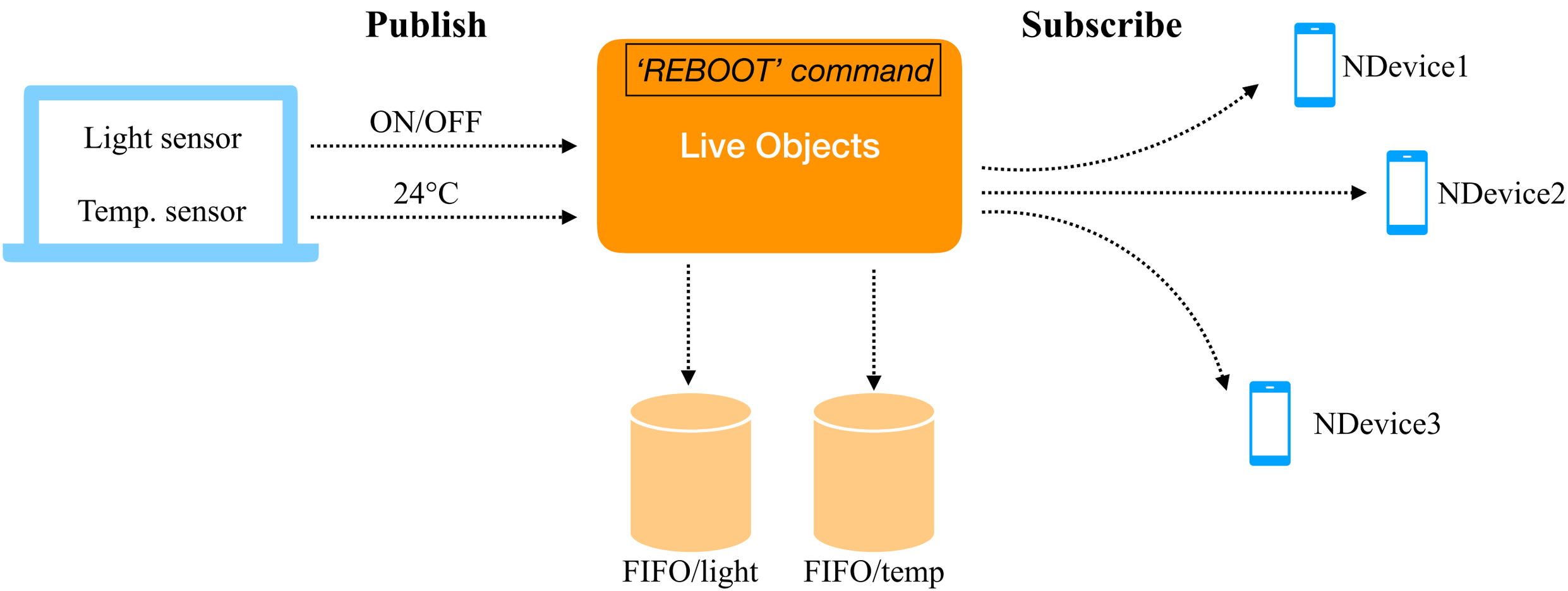
A Sever can: - Accepts network connections from Clients.
- Accepts messages published by Clients.
- Processes Subscribe and unsubscribe requests from Clients.
- Forwards messages that match Client subscriptions.

TOPIC: It is a label attached to the message which is matched against the Subscriptions known to the Server. During a subscription Clients can use filters to indicate an interest in one or more topics.



Ref: [http://mqtt.org/ (10/2018)]

Application Design



Connection Parameters

WiFi connection

SECRET_SSID: "Honor10"
SECRET_PASS: “xxxxxxx”

MQTT connection

USERNAME1: "json+device" //device mode connection + json encoding format
USERNAME: "json+bridge" //bridge mode connection + json encoding format

API_KEY: "fb38f68247524df1a17cdb99d3b46403"

SERVER: "liveobjects.orange-business.com"
PORT1 8883 //SSL connection
PORT2 1883

USER_ID: "urn:lo:nsid:ARDUINO_MKR_WIFI_1010:24:0A:C4:AD:E0:2C"

Device Mode

In ‘Device Mode’ the device has only a single MQTT connection with Live Objects. It is used to:

- To notify device connectivity status
- To notify and to received configuration update
- To received and to send commands
- To send data

For the connection the username credential has to be ‘json+device’ (json format + device mode)

Main topic and their utility

publish data	dev/data
receive a command	dev/cmd
replay to a command	dev/cmd/res
announce current configuration	dev/cfg
subscribe for receiving new configuration	dev/cfg/res

Bridge Mode

In ‘Bridge Mode’ a single MQTT connection is used to exchange data with multiple devices. It can be used when a device acts like gateway because it has to collects and exchange data with multiple devices.

Username credential can be ‘json+bridge’ (json format + bridge mode) or ‘payload+bridge’ (no encoding format + bridge mode)

In bridge mode topics where to publish and to subscribe have the following format:
fifo/{fifo name}
pubsub/{pubsub topic name}
router/{routingKey}

examples: topic name: *router/android/123/data* routing key filter: *android.123.data*

Routing Rules

- every / is replaced by a .
- every MQTT wildcard + is replaced by a *
- MQTT wildcard # stays #

Main device

1. connect in *device mode* (username: json+device)
2. subscribe to *dev/cmd* topic to received command from Live Objects
3. publish to *dev/cmd/res* topic to replay to the command received

4. connect *bridge mode* (username: json+bridge)
5. get data from light and temperature sensors
6. publish light data into *fifo/light* topic
7. publish temperature data into *fifo/temp* topic

NDevices

1. connect in *bridge mode* (username: json+bridge)
2. subscribe to topic *fifo/light*
3. received all messages inside the queue
4. subscribe to topic *fifo/temp*
5. received all messages inside the queue