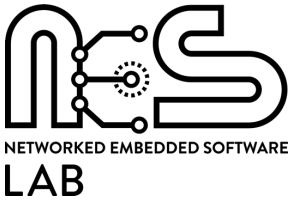
POLITECNICO
MILANO 1863



Networking with Node-RED

Luca Mottola
luca.mottola@polimi.it

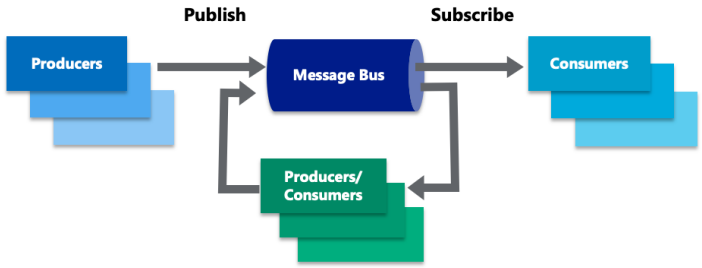
(version 0.1)

Outline

- About networking
- UDP sockets
- MQTT

MQTT

Publish/Subscribe



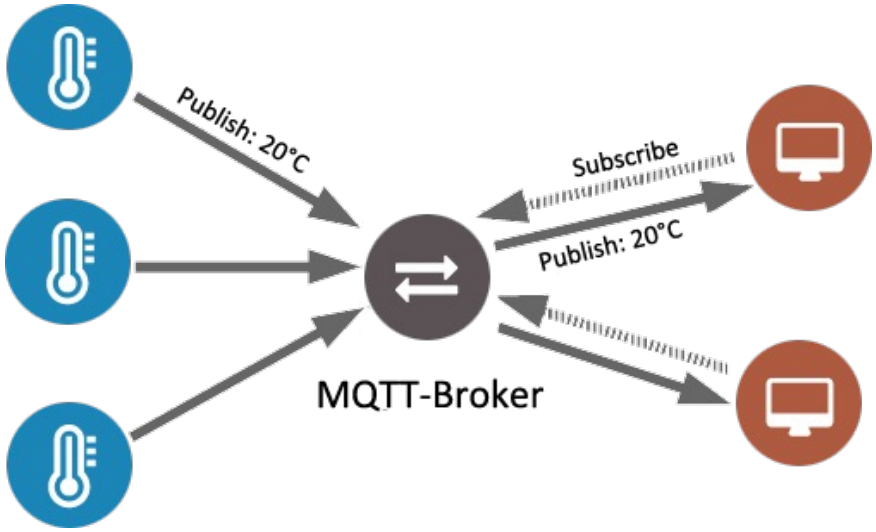
- Publish/Subscribe is group communication interaction pattern
  - **Subscribers** (**consumers**) express an interest in data
  - **Publishers** (**producers**) generate data that is disseminated in a message bus
- Subscriptions may be expressed based on
  - **Topics**: like a “channel”, group data with a common nature or shared features
    - Example: subscribe to all temperature messages
  - **Content**: pattern matching on the message content
    - Example: messages reporting temperature data above 20C in room ABC
- The **message broker** matches published data with existing subscriptions

Why Pub/Sub?

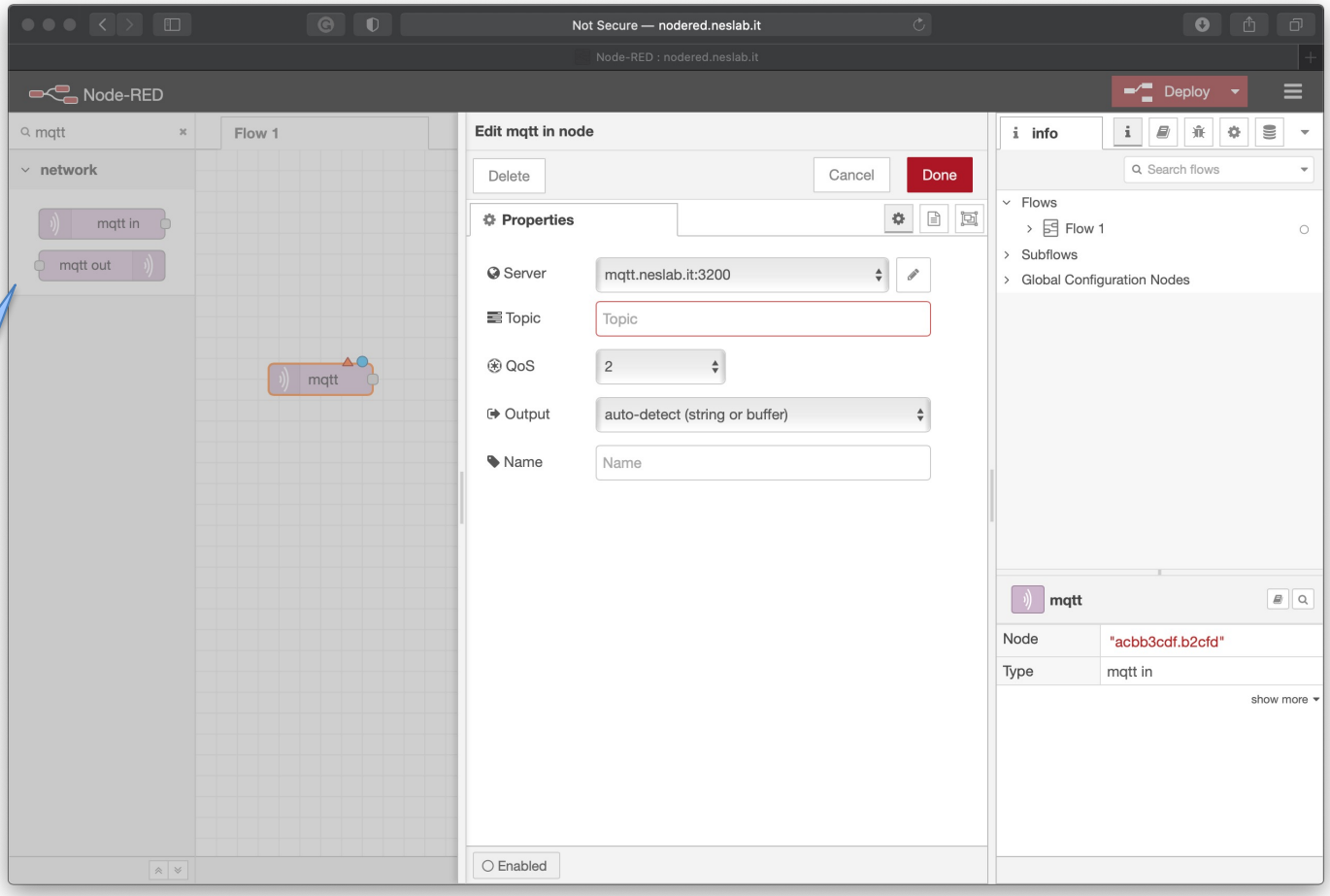
- **Decouples** data producers and consumers
  - They don't need to know each other!
  - No destination IP and port information needed
  - May come and go freely
- Communication becomes **data-centric**, rather than address-centric!
- …but, you need the broker!

MQTT Overview

- An extremely lightweight Pub/Sub messaging layer
- Payload agnostic
- **Topic-based**
  - Topics are arranged **hierarchically**
  - Examples:
    - `iot`
    - `iot/building21`
    - `iot/building21/temperature`
  - Effectively represent nested “channels”

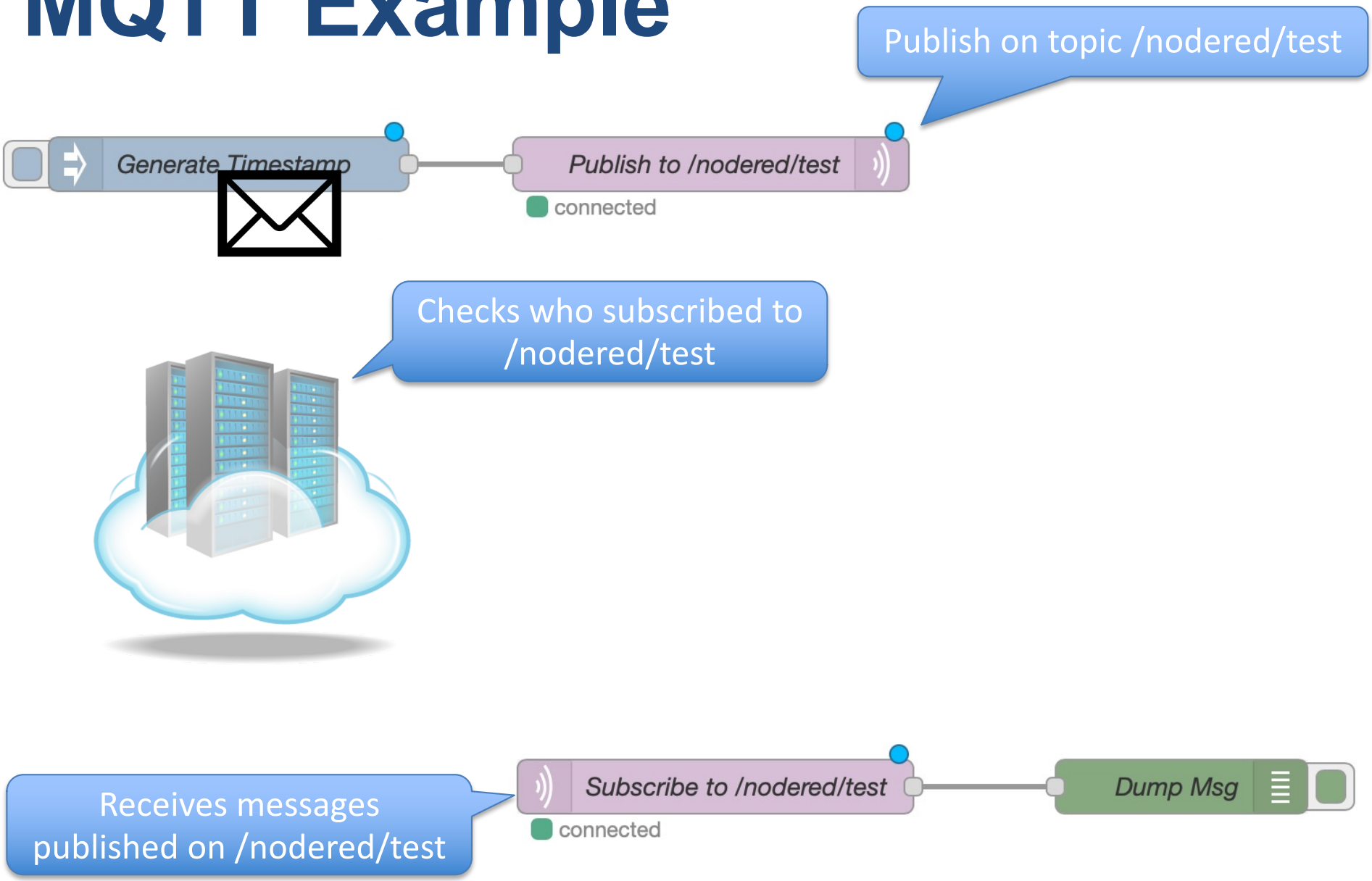


MQTT

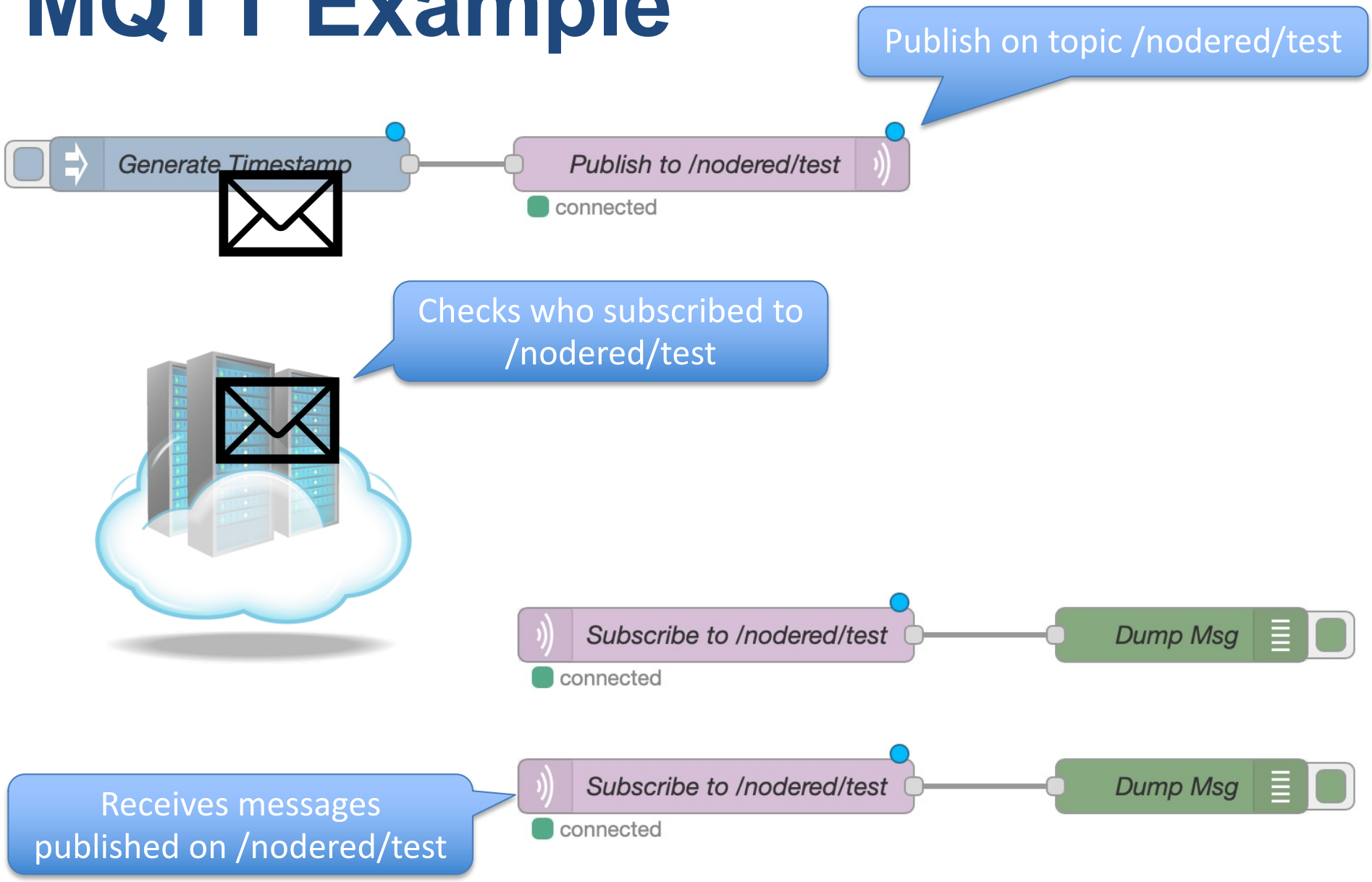


Nodes are provided to use MQTT as an input by expressing subscriptions, or as an output to publish messages

MQTT Example

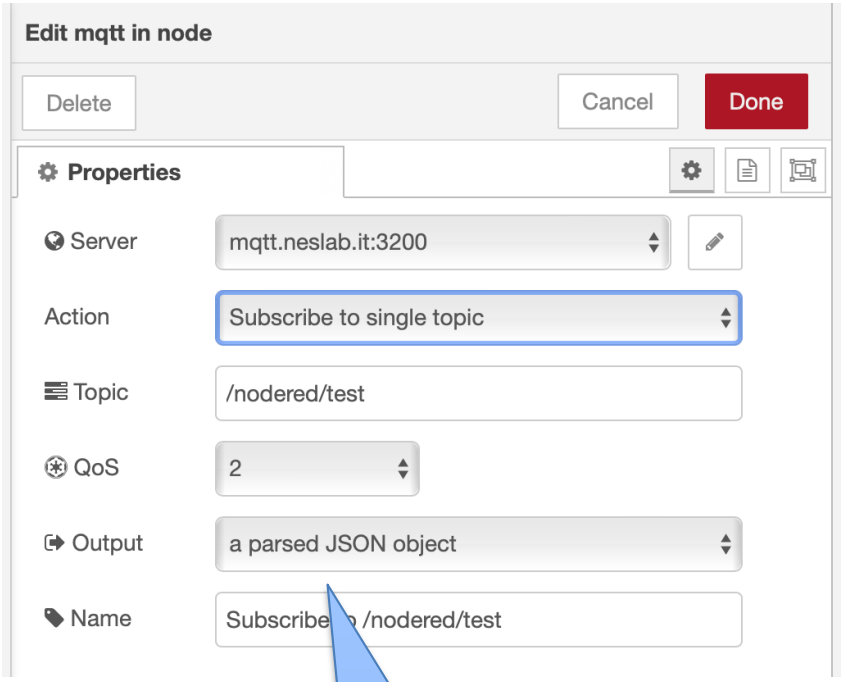


MQTT Example



Serialization in MQTT

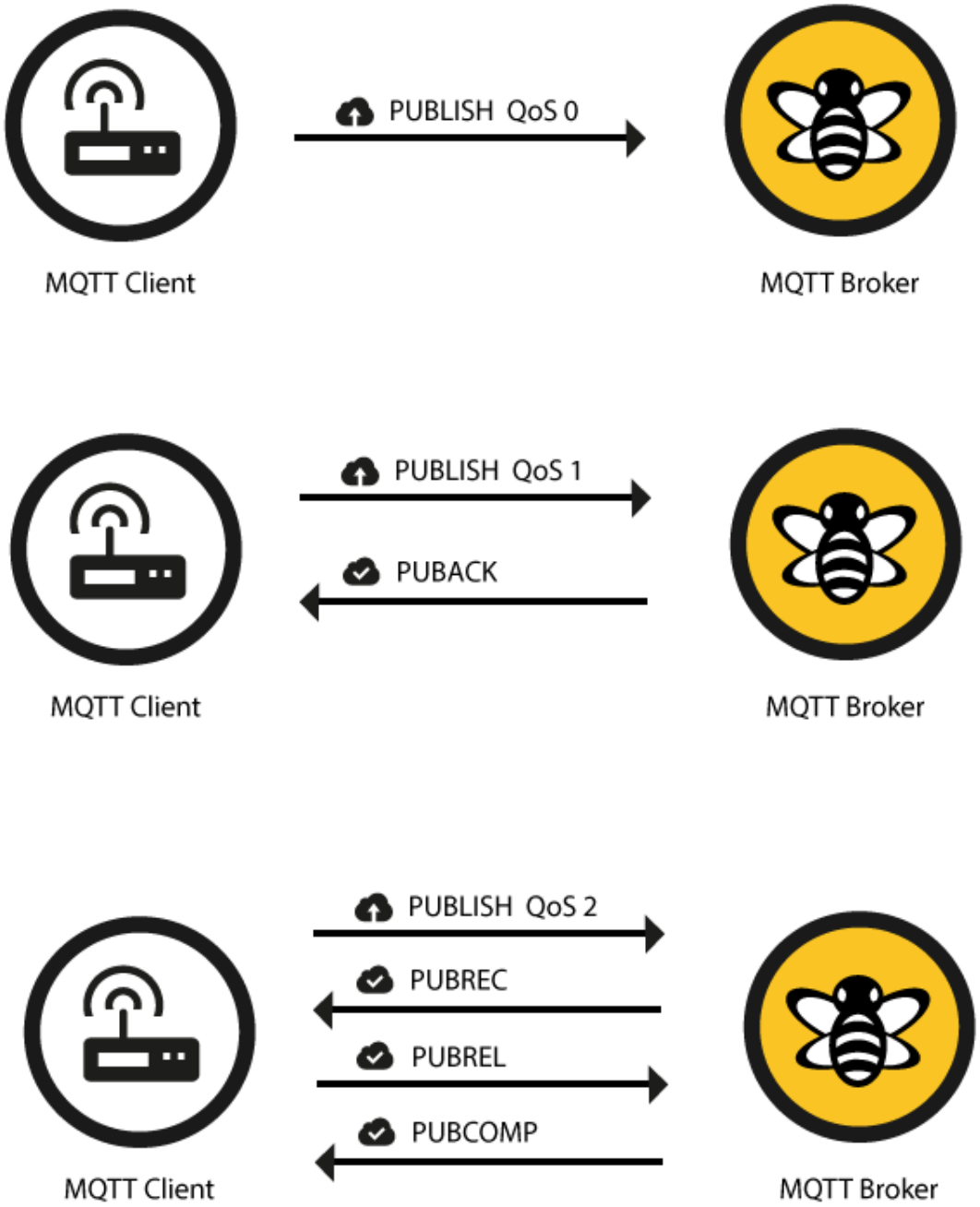
- MQTT handles messages as **Strings**
- When publishing, JavaScript objects are automatically **converted to JSON**
  - No need for an explicit JSON conversion node
- When receiving from a matching subscription, the output may be **converted directly to a JavaScript object**



Messages will be output as JavaScript objects in the payload

MQTT QoS

- MQTT QoS is a per-message agreement on the guarantees on delivery
- **QoS0**: at most once
- **QoS1**: at least once
- **QoS2**: at most once

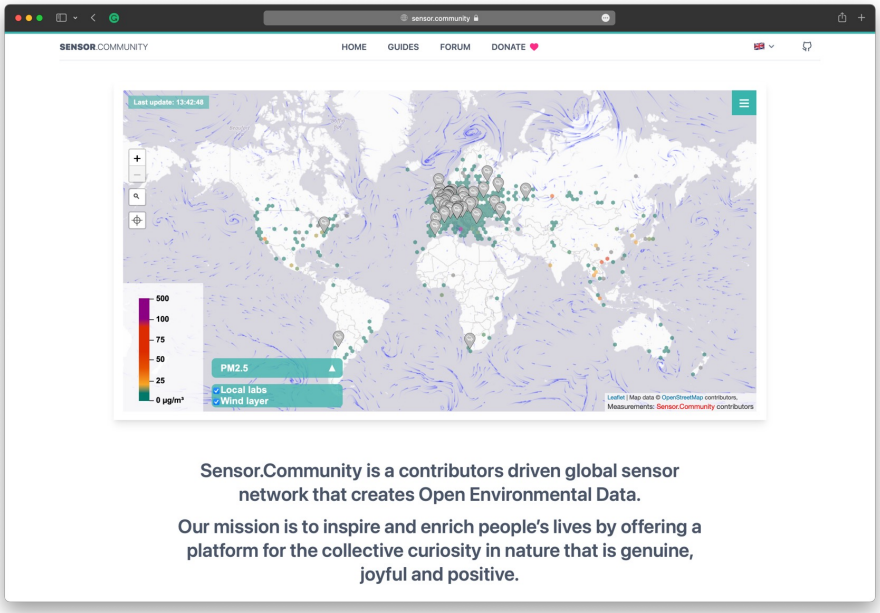


MQTT Persistent Sessions

- Normal behavior:
  - Clients subscribe upon connecting to the broker
  - Subscriptions are lost when disconnecting
- **Persistent sessions** save the subscriptions of clients as they go off-line
- If so, the MQTT broker **queues** all QoS1 and QoS2 messages published while that client was off-line
- The client automatically **re-send** unconfirmed QoS1 and QoS2 published messages
- **Careful**: may backfire if the client is long disconnected

MQTT For Integration

- MQTT is often used as a basis for integrating different data sources
- As long as a system can publish over MQTT, data can reach everywhere
- Example: publishing **`sensor.community`** over MQTT



MQTT: Sensor.Community