

Fundamentals of IoT Software © 2022 by Luca Mottola  
is licensed under CC BY-NC 4.0



To view a copy of this license, visit  
[creativecommons.org/licenses/by-nc/4.0/](https://creativecommons.org/licenses/by-nc/4.0/)

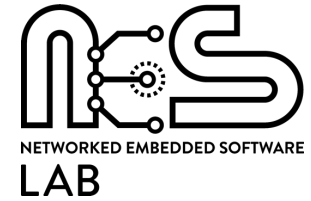




**POLITECNICO**  
MILANO 1863



**POLITECNICO**  
MILANO 1863



# Networking with Node-RED

**Luca Mottola**

`luca.mottola@polimi.it`

(version 0.1)

# Outline

- About networking
- UDP sockets
- MQTT

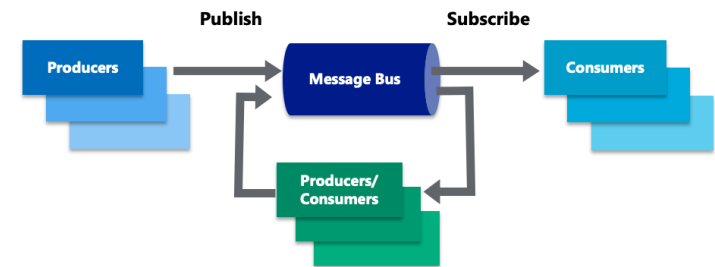


# MQTT



# 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



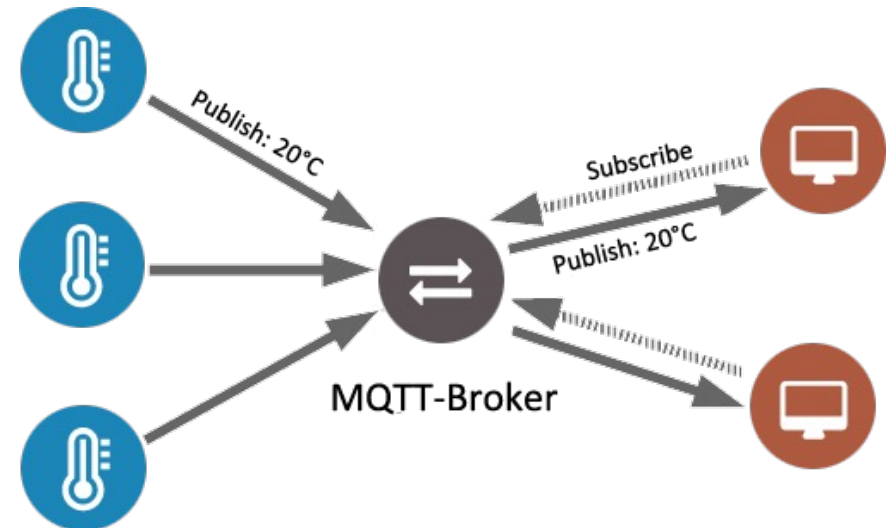
# Why Pub/Sub?

- **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

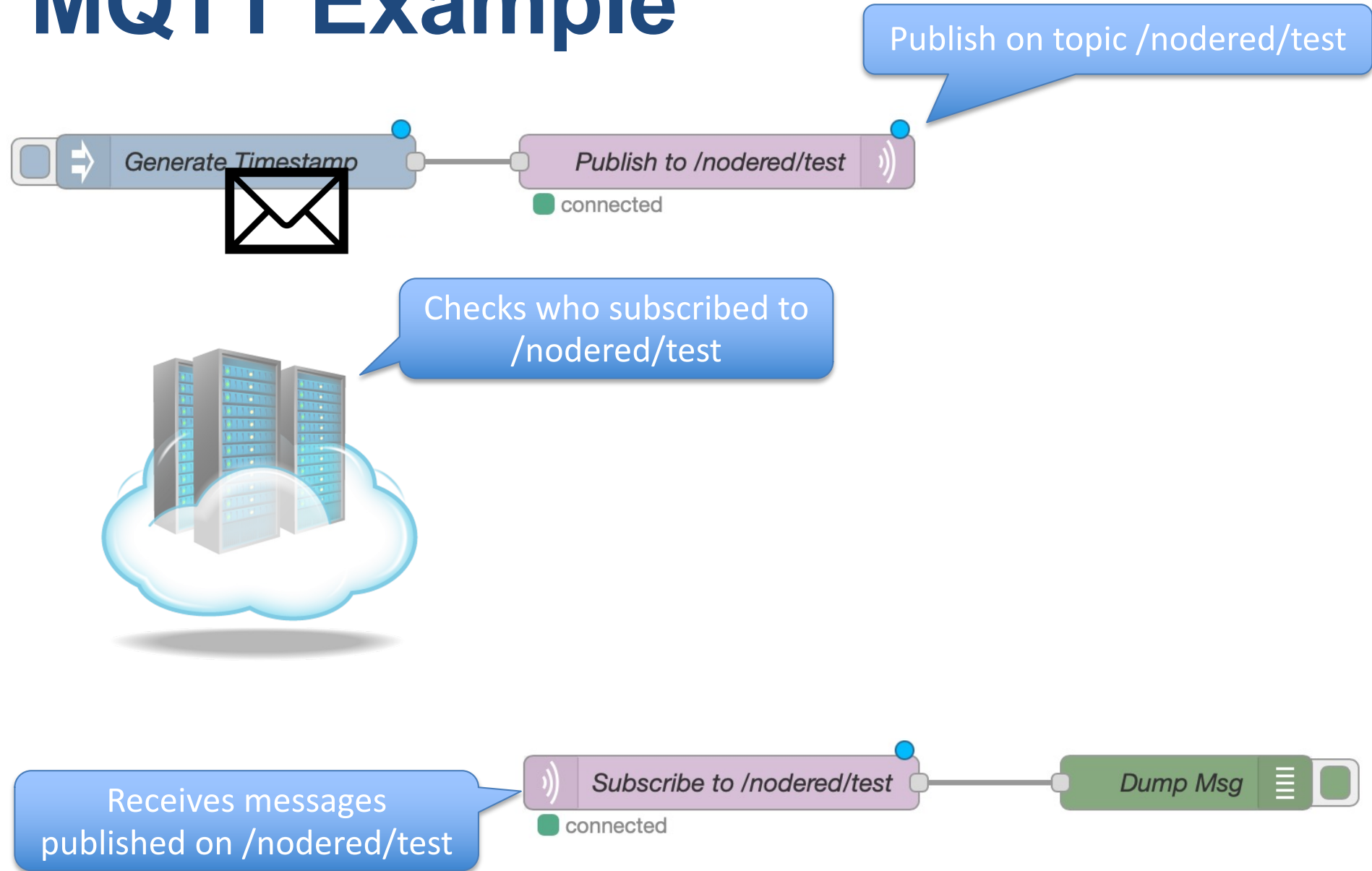
The screenshot displays the Node-RED web interface in a browser window titled "Not Secure — nodered.neslab.it". The main workspace shows a flow named "Flow 1" with three MQTT nodes: "mqtt in", "mqtt out", and a generic "mqtt" node. The left sidebar contains a search bar with "mqtt" entered and a list of nodes under the "network" category. The right sidebar is divided into two panels. The top panel, titled "Edit mqtt in node", contains configuration fields: "Server" (mqtt.neslab.it:3200), "Topic" (Topic), "QoS" (2), "Output" (auto-detect (string or buffer)), and "Name" (Name). The bottom panel, titled "info", shows a tree view of the flow and a table for the selected "mqtt" node.

Node	Type
"acbb3cdf.b2cfd"	mqtt in



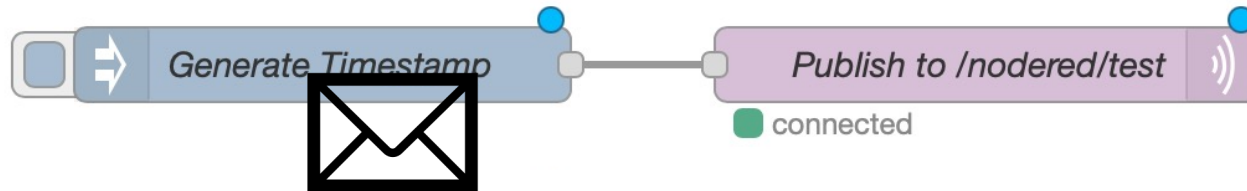


# MQTT Example

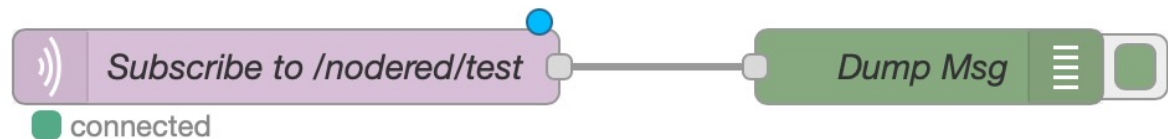


# MQTT Example

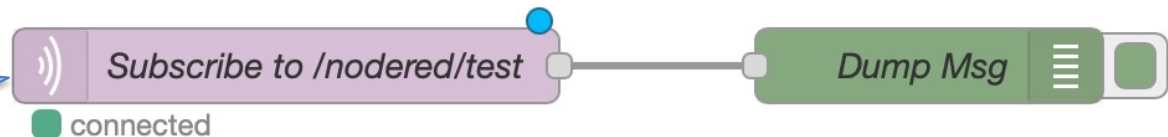
Publish on topic /nodered/test



Checks who subscribed to /nodered/test



Receives messages published on /nodered/test



# 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**

Edit mqtt in node

Delete Cancel Done

Properties

Server mqtt.neslab.it:3200

Action Subscribe to single topic

Topic /nodered/test

QoS 2

Output a parsed JSON object

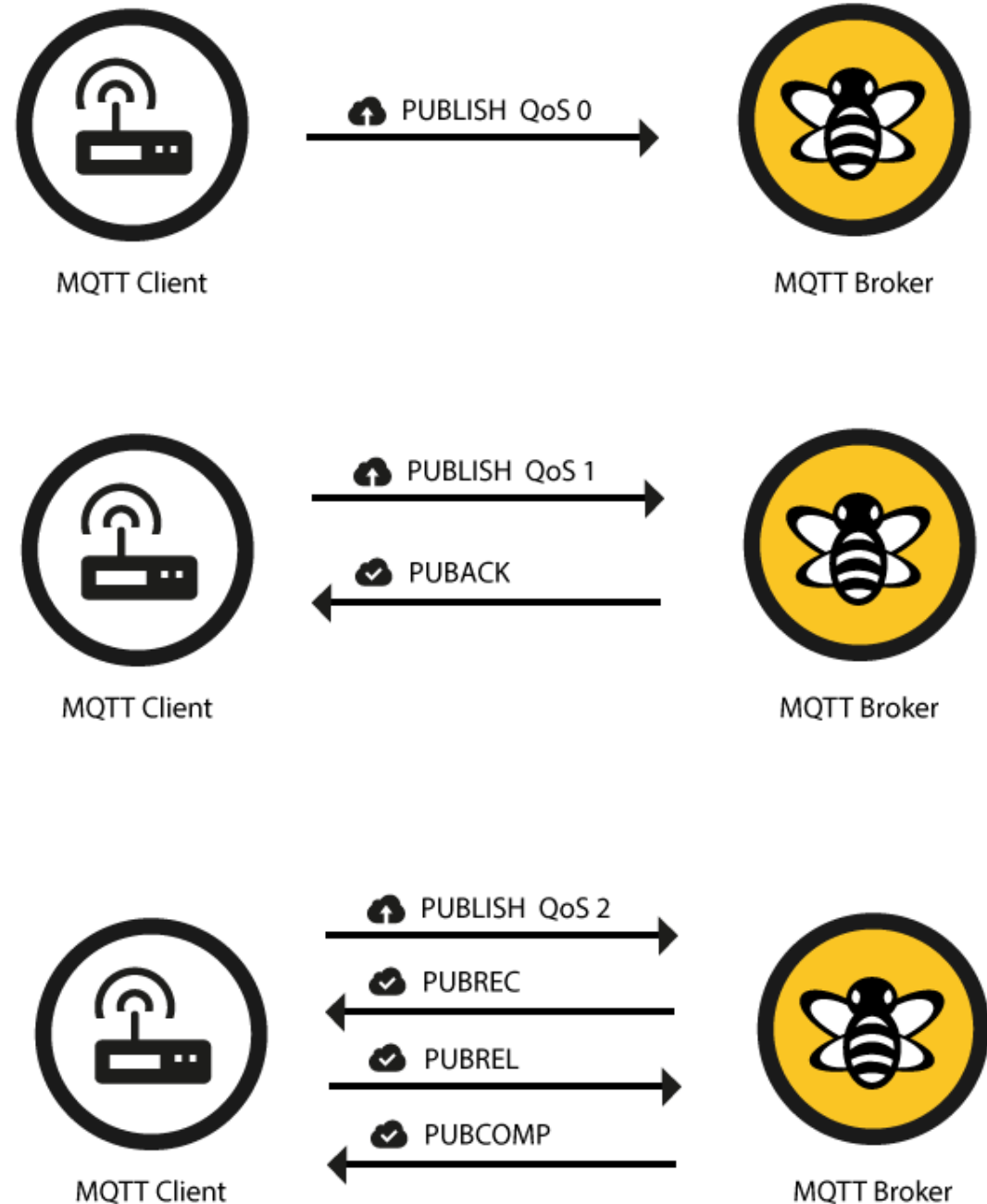
Name Subscribe to /nodered/test

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



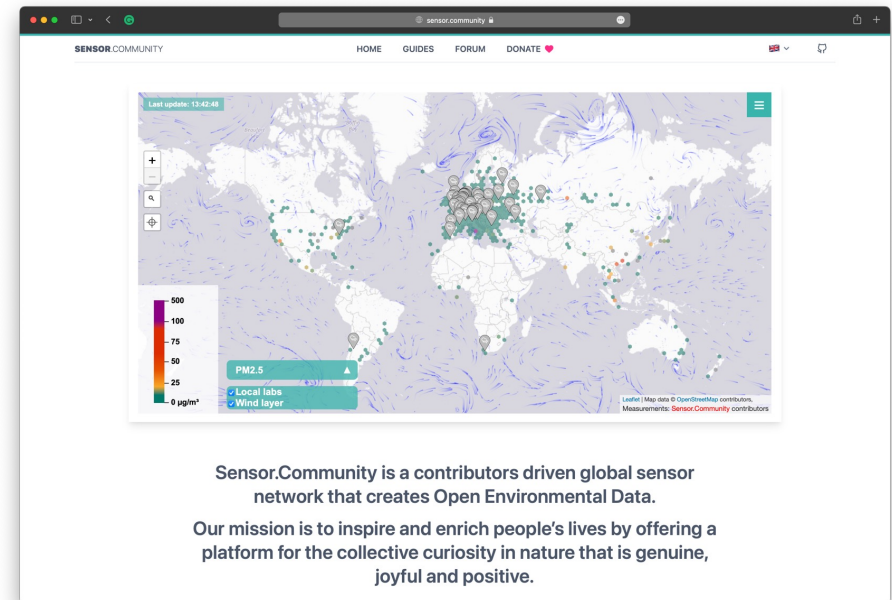
# MQTT Persistent Sessions

- 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

```
▼ object
  topic: "/smartcity/milan"
▼ payload: object
  id: 10372898440
  timestamp: "2022-05-06 11:46:13"
▼ sensordatavalues: array[2]
  ▼ 0: object
    id: 22999226740
    value_type: "temperature"
    value: "16.90"
  ▼ 1: object
    id: 22999226768
    value_type: "humidity"
    value: "99.90"
▼ sensor: object
  id: 21666
  pin: "7"
▼ sensor_type: object
  id: 9
  name: "DHT22"
  manufacturer: "various"
▼ location: object
  altitude: "125.4"
  latitude: "45.486"
  id: 10994
  indoor: 0
  country: "IT"
  exact_location: 0
  longitude: "9.19"
  sampling_rate: null
qos: 0
retain: false
_msgid: "56cdcde9741bc384"
```

Timestamp

Physical quantities and values

Information on sensing hardware

Geolocalization

MQTT parameters

