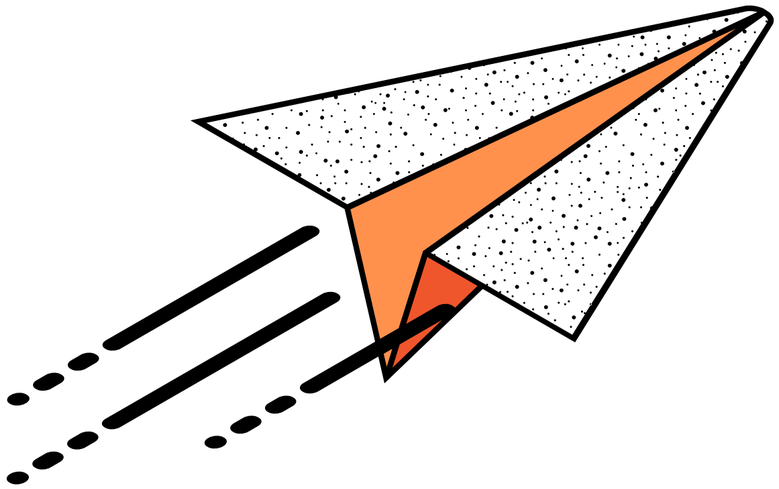
borda
IoT for Healthcare

MQTT 101

Introduction to MQTT



- What is MQTT?
- Main Features & Under the Hood
- Basic Concepts

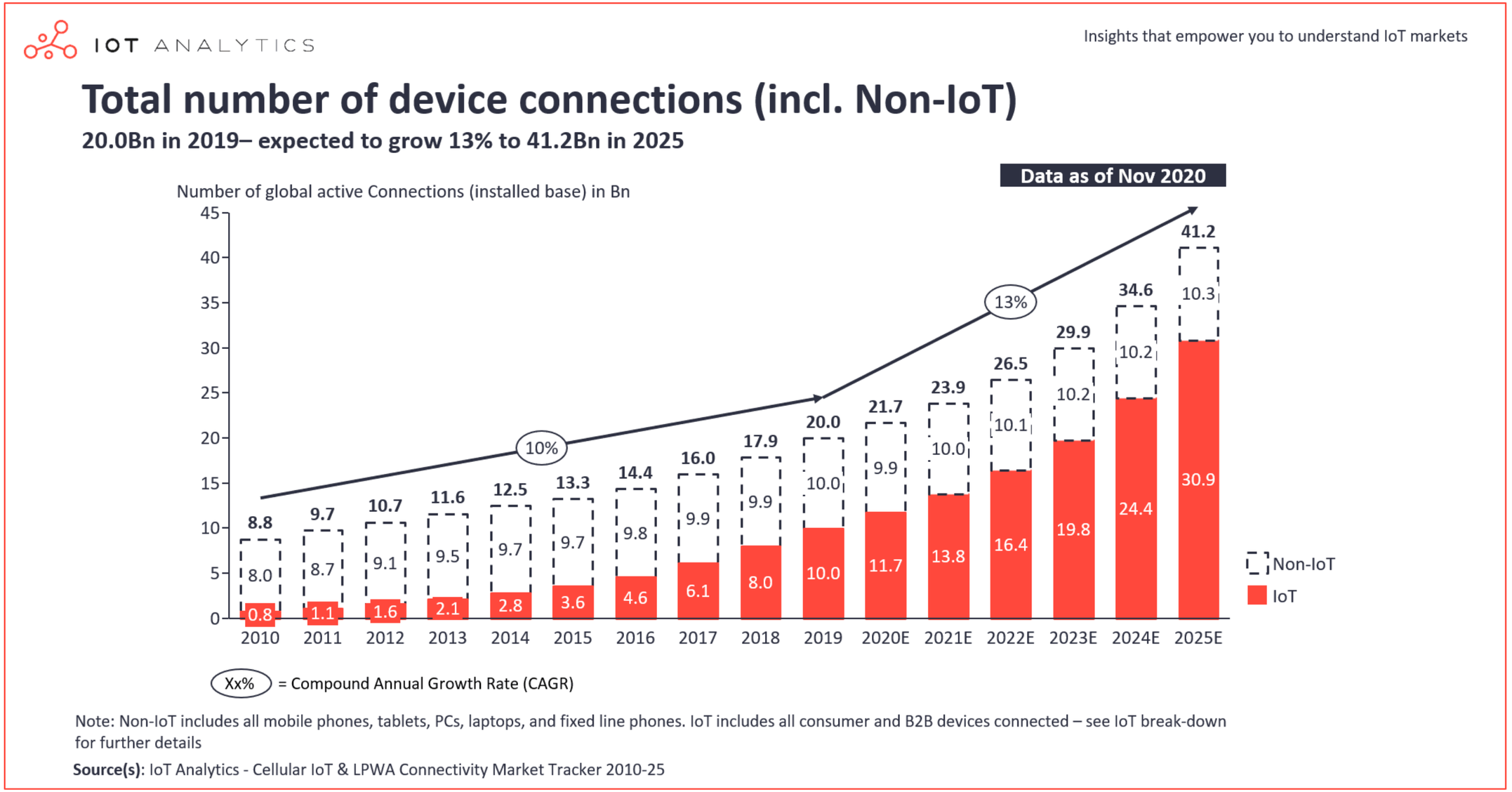
40 Min

- Topics and Architucture
- Features
- MQTT 3.1.1 vs MQTT 5 vs AMQP

60 Min

- Demo

16 Billion IoT Devices - 8 Billion MQTT Devices



https://www.globenewswire.com/en/news-release/2021/12/07/2347241/0/en/The-Eclipse-Foundation-Releases-2021-IoT-Edge-Developer-Survey-Results.html

https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/

"MQTT is a **publish/subscribe** messaging transport protocol. It is **light weight**, **open**, **simple**, and designed so as to be **easy to implement**. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in **Machine to Machine** (M2M) and **Internet of Things** (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium."

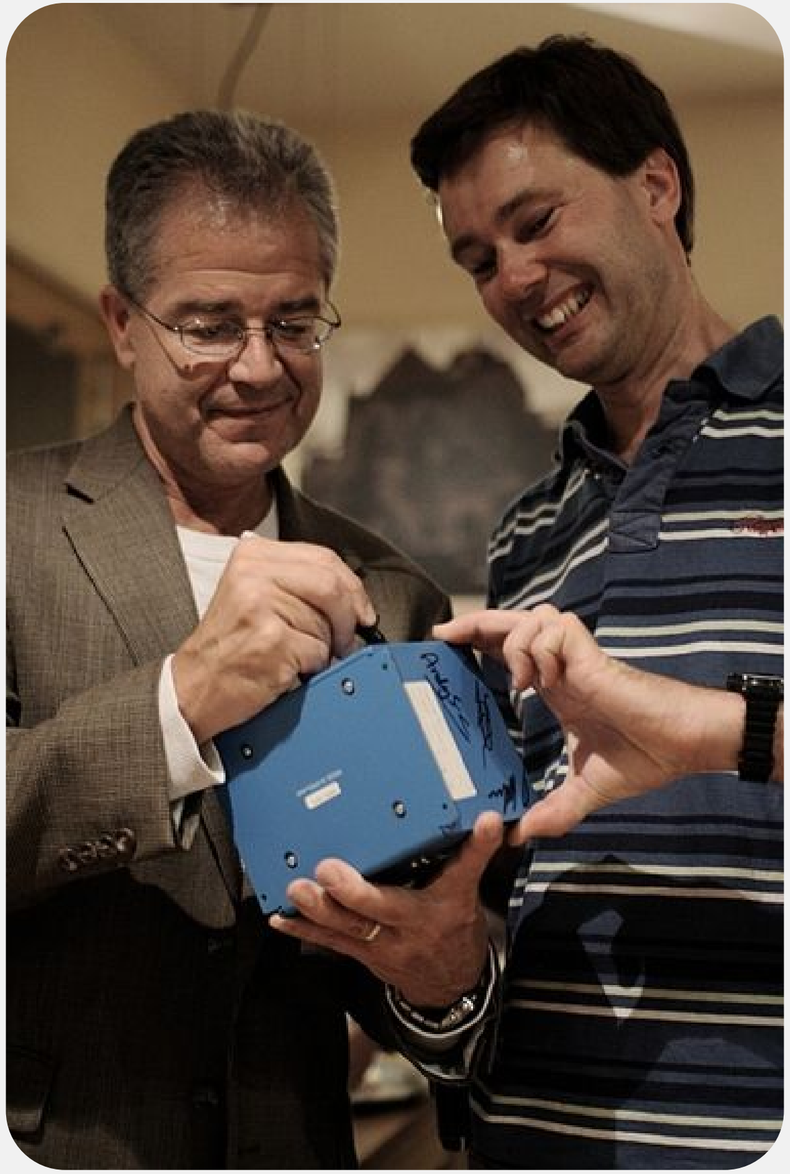
Citation from the official MQTT 3.1.1 specification

borda
IoT for Healthcare

History of MQTT

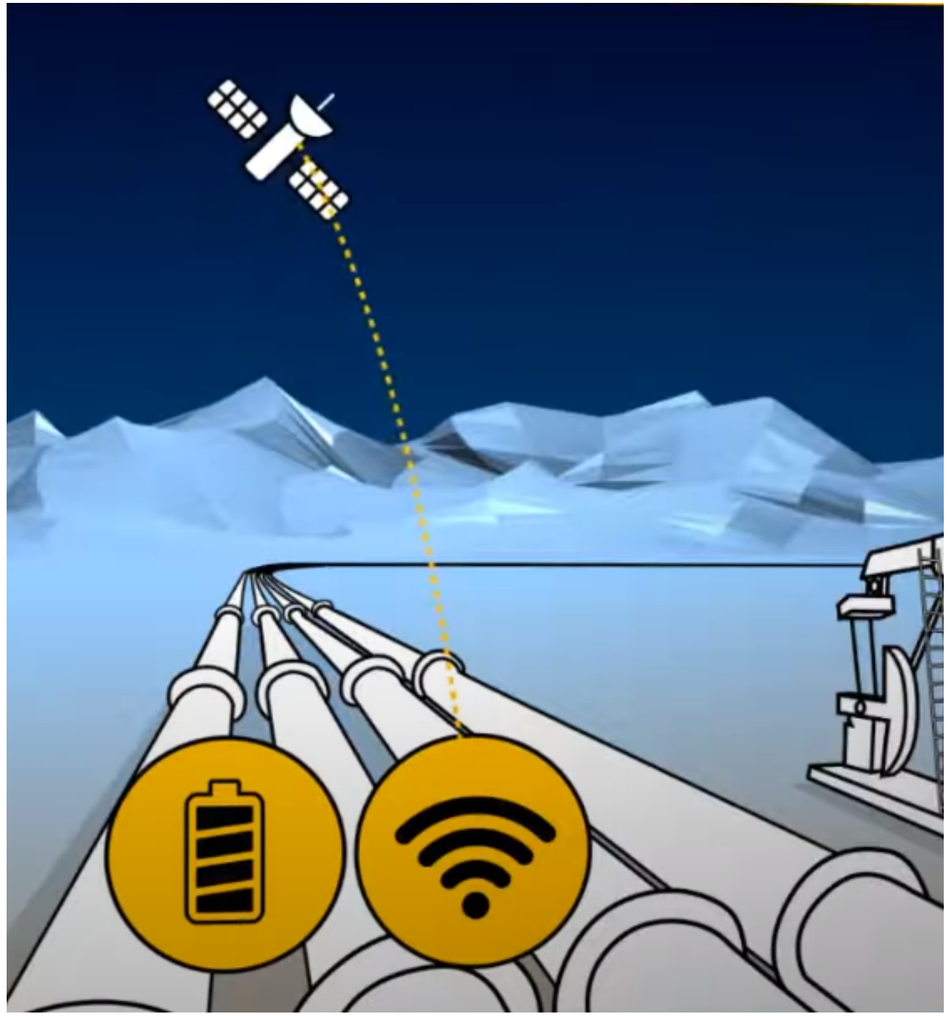
MQ Telemetry Transport

The MQTT protocol was invented in 1999 by Andy Stanford-Clark (IBM) and Arlen Nipper.



Requirements

1 - Simple Implementation
2 - Quality of Service Data Delivery
3 - Lightweight and Bandwidth Efficient
4 - Data-Agnostic
5 - Continous Session Awareness



borda
IoT for Healthcare

History of MQTT

1

1999

MQTT
Invention

2

2010

MQTT 3.1
Royalty-Free Release

3

2014

MQTT 3.1.1
OASIS Standard

4

2016

MQTT 3.1.1
ISO Standart

5

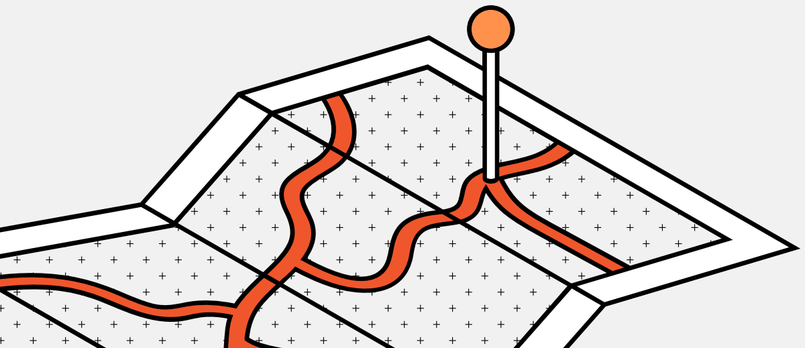
2018

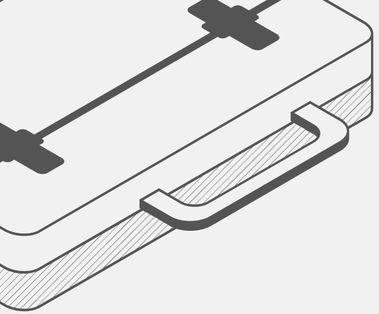
MQTT 5
Initial Release

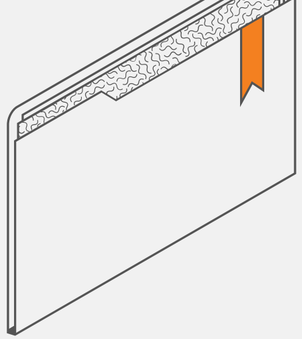
6

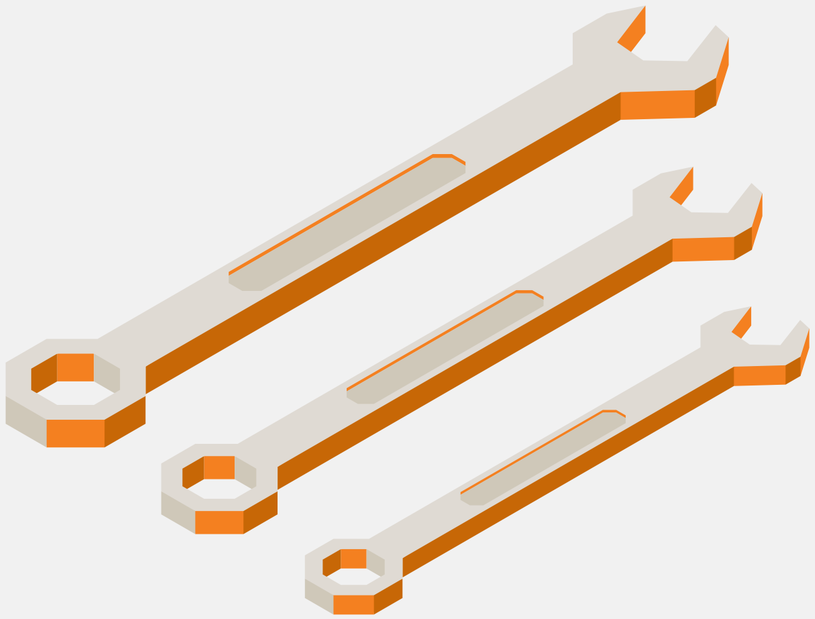
2019

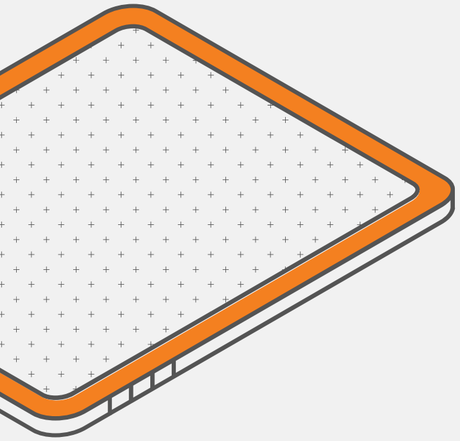
MQTT 5
OASIS Standard











Main Features & Under the Hood

borda
IoT for Healthcare

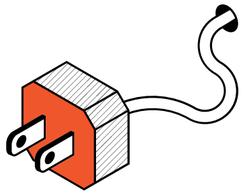
Main Features

Main Features

IoT Messaging Protocol
On Top of TCP
Minimal Overhead
Simple
Designed for Reliable Communications over Unreliable Channels.

Characteristics

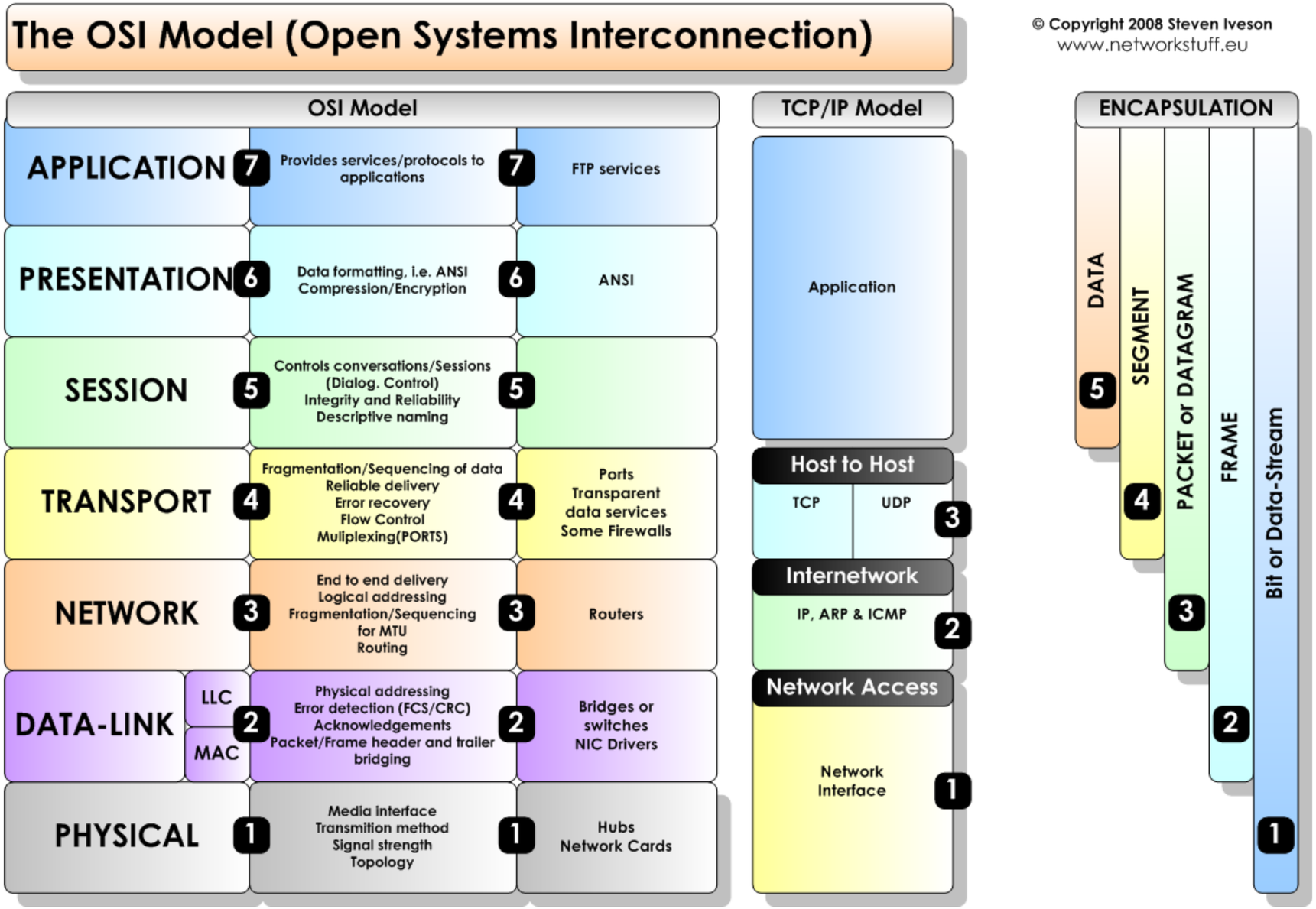
Efficient
Bi-Directional
Data-Agnostic
Scaleable
Built for Push Communication
Suitable for Constrained Devices

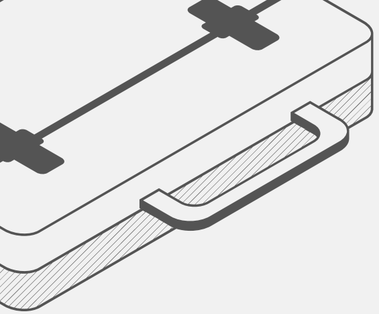


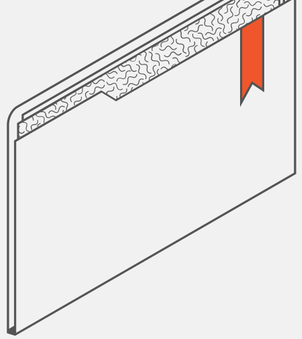
borda
IoT for Healthcare

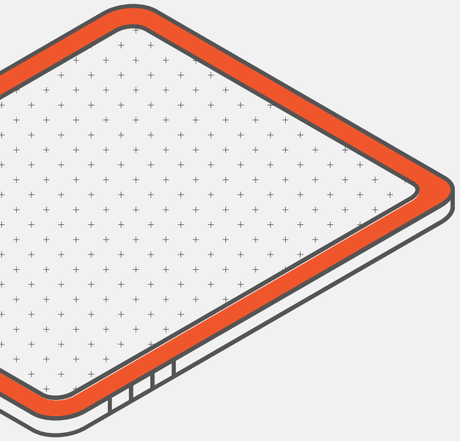
Under the Hood

MQTT	AMQP	HTTP	CoAP	DDS
TCP				UDP
IP				
PHY/MAC: BLE, Wi-Fi, 3GPP, etc.				







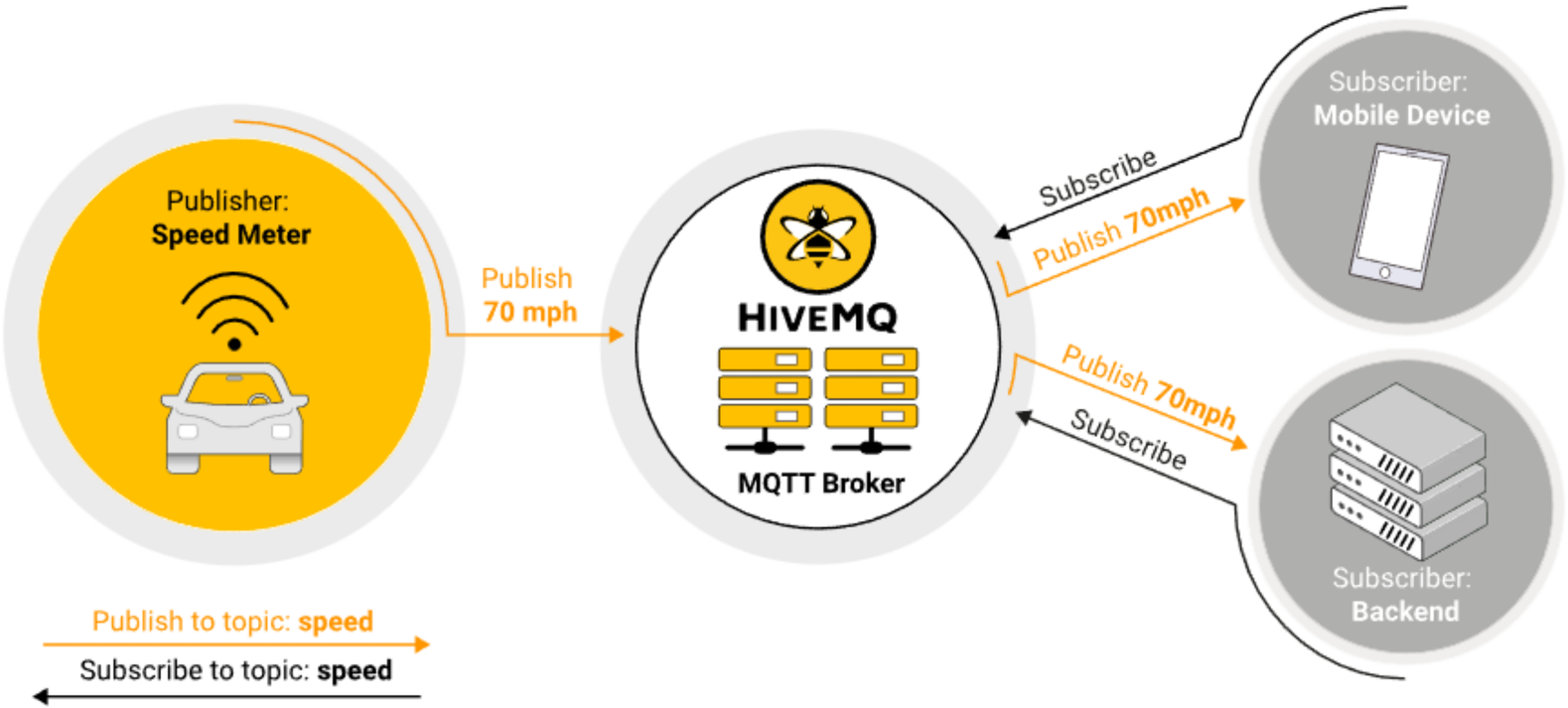


Basic Concepts

Publish/Subscribe Basics
Client/Broker Basics
Publish, Subscribe and Unsubscire Packets

Publish/Subscribe Basics

The publish/subscribe pattern (also known as pub/sub) provides an alternative to traditional client-server architecture.



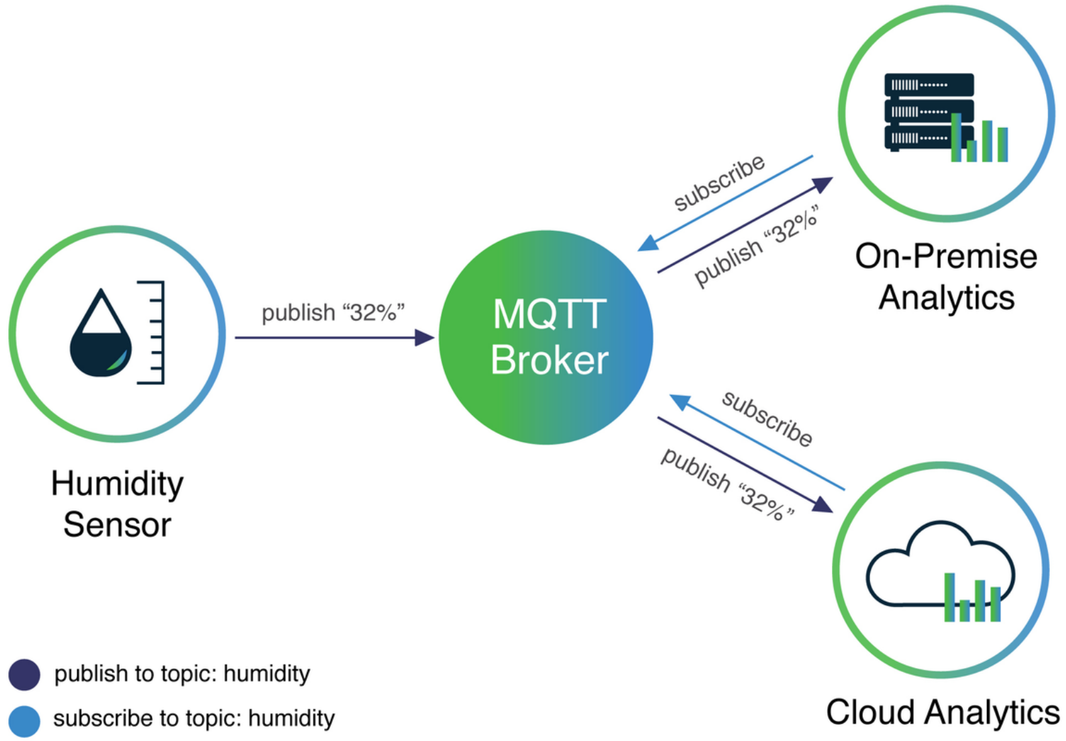
Publish/Subscribe Basics

Space decoupling: Publisher and subscriber do not need to know each other (for example, no exchange of IP address and port).

Time decoupling: Publisher and subscriber do not need to run at the same time.

Synchronization decoupling: Operations on both components do not need to be interrupted during publishing or receiving.

Single Point of Failure (SPOF)

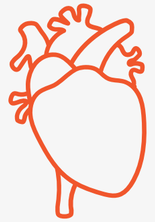


Client/Broker Basics

Client

When we talk about a client, we almost always mean an MQTT client. Both **publishers** and **subscribers** are MQTT clients. The publisher and subscriber labels refer to whether the client is currently publishing messages or subscribed to receive messages.

Broker

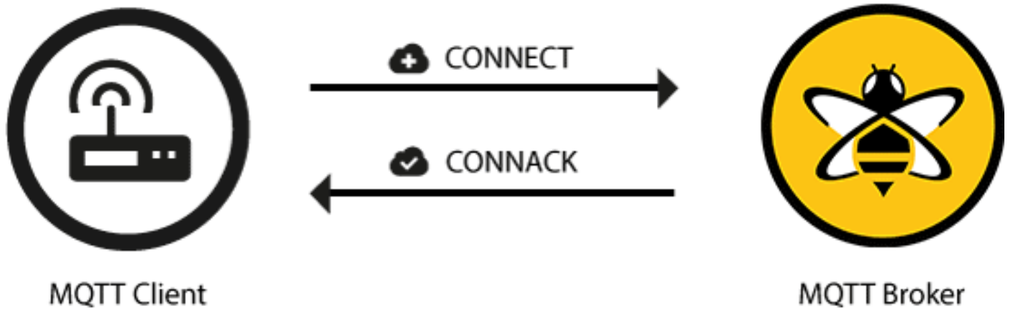


The broker is at the **heart** of any publish/subscribe protocol.

The broker is responsible for receiving all messages, **filtering the messages**, determining who is subscribed to each message, and sending the message to these subscribed clients. The broker also holds the session data of all clients that have persistent sessions, including subscriptions and **missed messages**

Connections

To initiate a connection, the client sends a CONNECT message to the broker. The broker responds with a CONNACK message and a status code.



MQTT-Packet:

CONNECT

contains:	Example
clientId	"client-1"
cleanSession	true
username (optional)	"hans"
password (optional)	"letmein"
lastWillTopic (optional)	"/hans/will"
lastWillQos (optional)	2
lastWillMessage (optional)	"unexpected exit"
lastWillRetain (optional)	false
keepAlive	60

The client identifier (ClientId) identifies each MQTT client that connects to an MQTT broker. The broker uses the ClientId to identify the client and the current state of the client.Therefore, this **Id should be unique per client** and broker.

borda
IoT for Healthcare

Connections

MQTT-Packet:

CONNECT

contains:	Example
clientId	"client-1"
cleanSession	true
username (optional)	"hans"
password (optional)	"letmein"
lastWillTopic (optional)	"/hans/will"
lastWillQos (optional)	2
lastWillMessage (optional)	"unexpected exit"
lastWillRetain (optional)	false
keepAlive	60

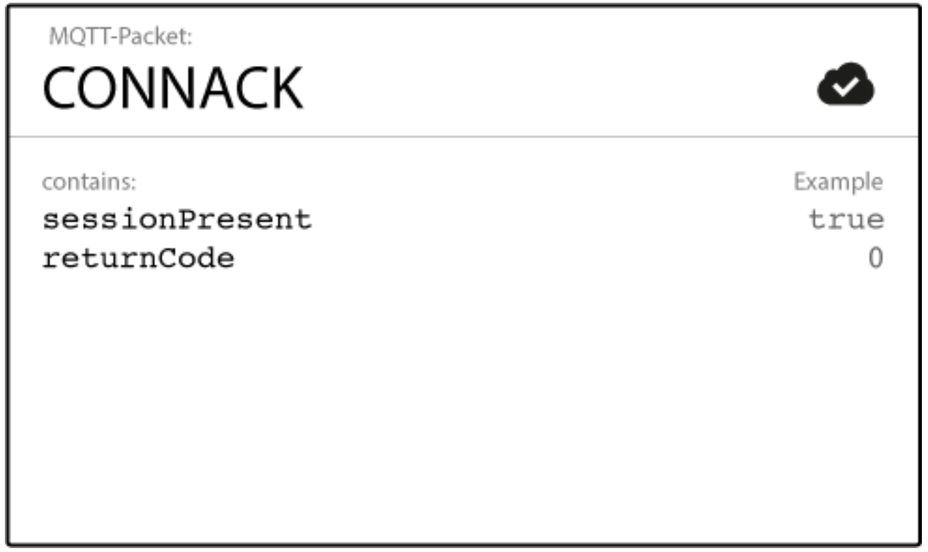
PINGREQ

MQTT can send a user name and password for client authentication and authorization

The keep alive is a time interval in seconds that the client specifies and communicates to the broker when the connection established. This interval defines the longest period of time that the broker and client can endure without sending a message.

borda
IoT for Healthcare

Connections



This flag contains a return code that tells the client whether the connection attempt was successful or not.

Return Code	Return Code Response
0	Connection accepted
1	Connection refused, unacceptable protocol version
2	Connection refused, identifier rejected
3	Connection refused, server unavailable
4	Connection refused, bad user name or password
5	Connection refused, not authorized

Publish, Subscribe & Unsubscribe

MQTT-Packet:

PUBLISH

contains:	Example
packetId (always 0 for qos 0)	4314
topicName	"topic/1"
qos	1
retainFlag	false
payload	"temperature:32.5"
dupFlag	false

Publish

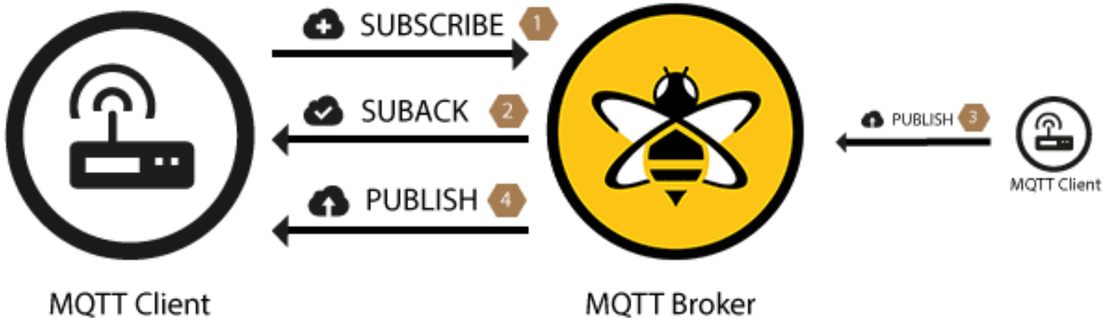
An MQTT client can publish messages as soon as it connects to a broker. MQTT utilizes topic-based filtering of the messages on the broker. Each message must contain a topic that the broker can use to forward the message to interested clients.

Publish, Subscribe & Unsubscribe

MQTT-Packet:

SUBSCRIBE

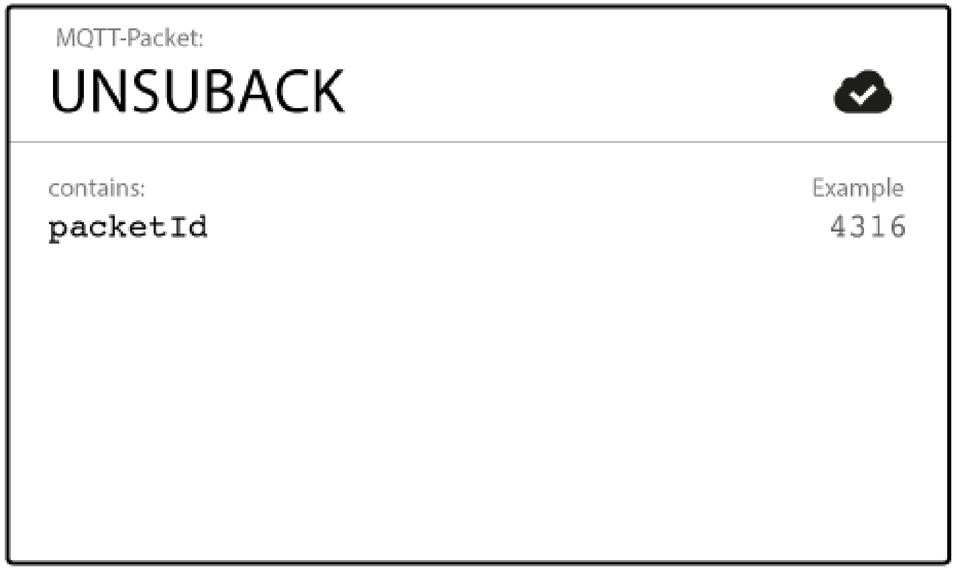
contains:		Example
packetId		4312
qos1	} (list of topic + qos)	1
topic1		"topic/1"
qos2	}	0
topic2		"topic/2"
...		...



Subscribe

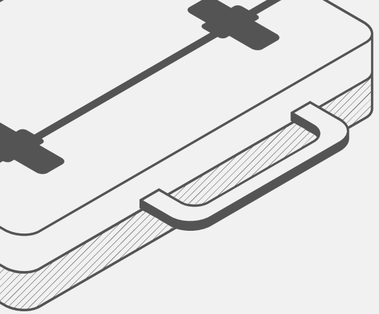
To receive messages on topics of interest, the client sends a SUBSCRIBE message to the MQTT broker. This subscribe message is very simple, it contains a unique packet identifier and a list of subscriptions.

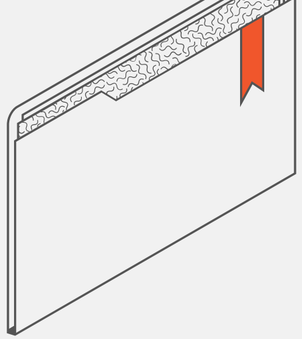
Publish, Subscribe & Unsubscribe

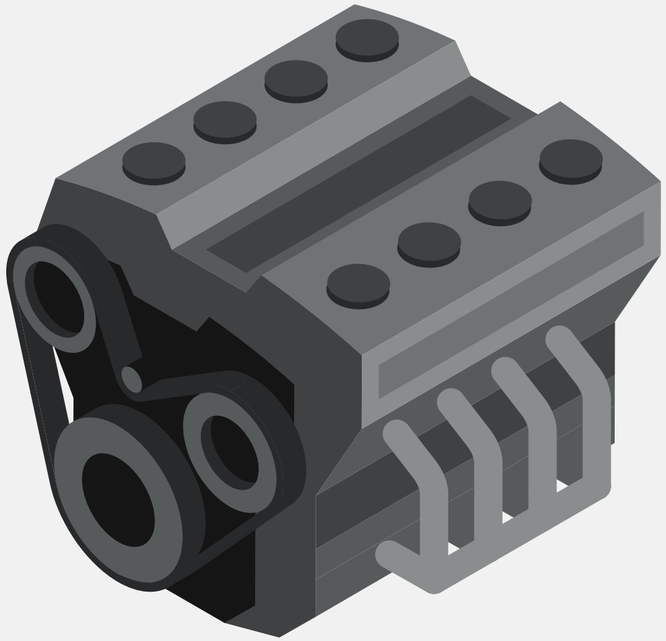


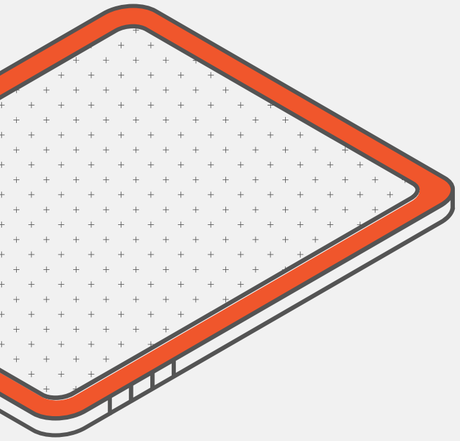
Unsubscribe

The counterpart of the SUBSCRIBE message is the UNSUBSCRIBE message. This message deletes existing subscriptions of a client on the broker. The UNSUBSCRIBE message is similar to the SUBSCRIBE message and has a packet identifier and a list of topics.









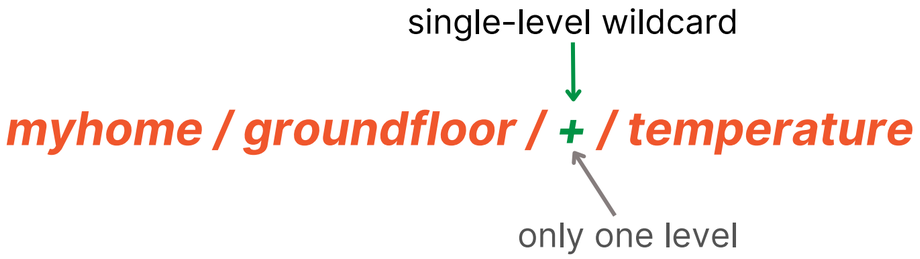
Topics and Architecture

borda
IoT for Healthcare

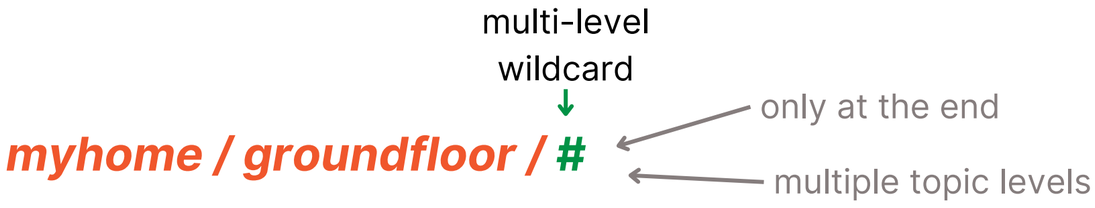
Topics

In MQTT, the word topic refers to an UTF-8 string that the broker uses to filter messages for each connected client.

- myhome / groundfloor / livingroom / temperature
- myhome / groundfloor / kitchen / temperature
- myhome / groundfloor / kitchen / brightness
- myhome / firstfloor / kitchen / temperature
- myhome / groundfloor / kitchen / fridge / temperature

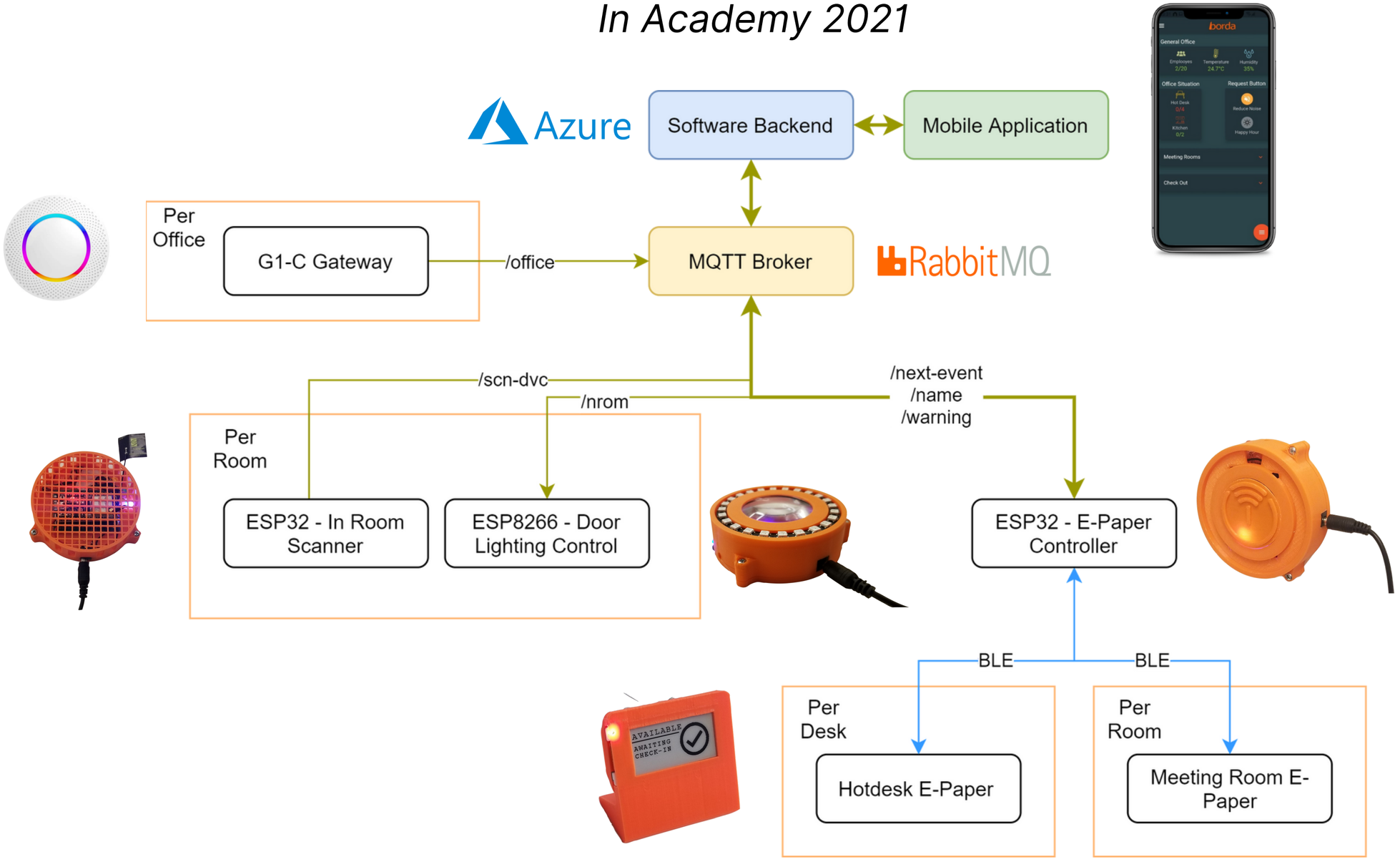


- myhome / groundfloor / livingroom / temperature
- myhome / groundfloor / kitchen / temperature
- myhome / groundfloor / kitchen / brightness
- myhome / firstfloor / kitchen / temperature



How we used Topics

In Academy 2021



/nrom

Subscribed by ESP8266
(Number of people in the room)

```
00:1B:44:11:3A:B7;4;10
Mac Address of ESP32;Number of people in the room;Capacity
```

Kitchen

Meeting Room

/scn-dvc

Published by devices ESP32
(Contains information about scanned devices)

```
{
   "e":[
      {
         "m":"ac:23:3f:a3:35:68",
         "r":"-55"
      },
      {
         "m":"d4:1c:38:db:d8:44",
         "r":"-77"
      }
   ],
   "mac":"84:0D:8E:2C:66:3C"
}
```

Kitchen

Meeting Room

/temp-hmd

Published by the Meeting Room and Kitchen LED ESP8266
Contains information about the name of the temperature and humidity value.

```
{
"mac": "00:1B:44:11:3A",
"temp": 34,
"hmd": 78
}
```

/name

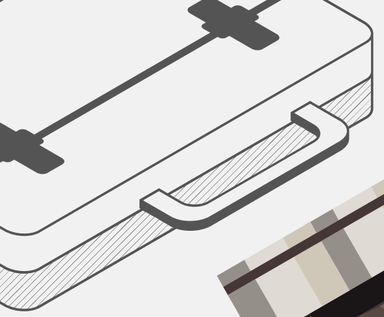
Subscribed by the Hotdesk ESP32s
Contains information about the name of the check-in person and Desk ID.

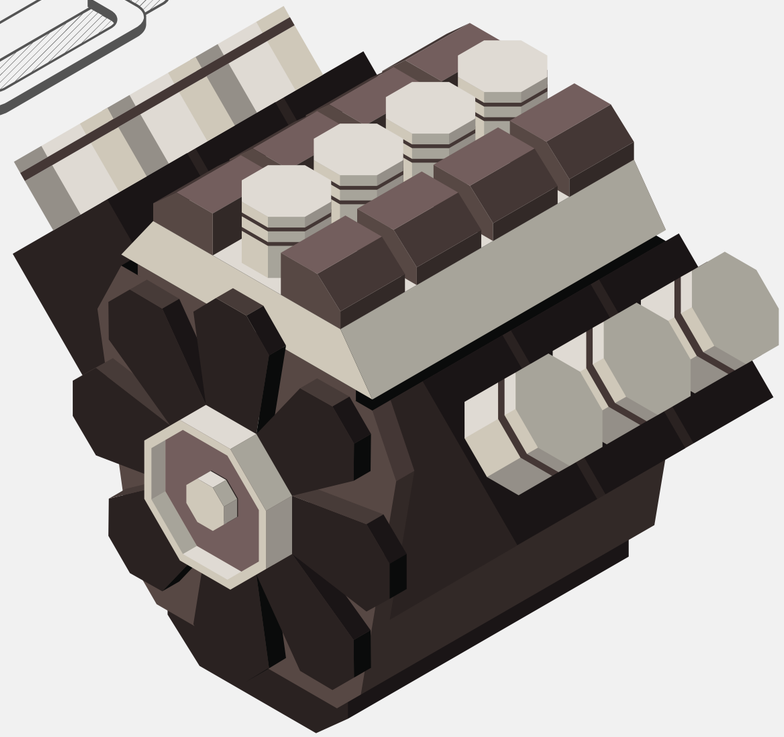

```
3C:71:BF:F5:5D:58;e24b0000-67f5-479e-8711-b3b99198ce6c;Ata Korkusuz
Mac Address of ESP32;Device UUID;Name
```

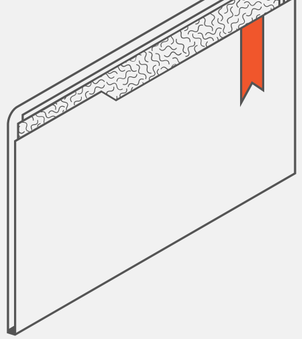
/next-event

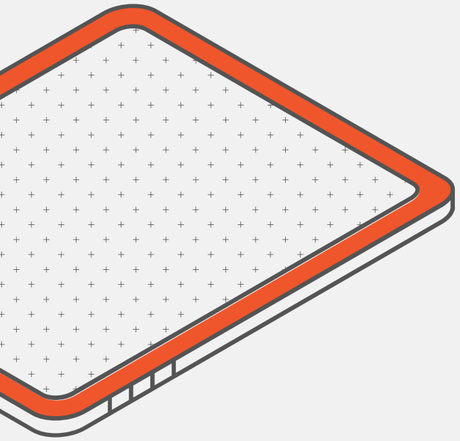
Subscribed by ESP8266 and ESP32s
(Contains information about meeting room or kitchen)

```
00:1B:44:11:3A:B7;e14b0000-67f5-479e-8711-b3b99198ce6c;Today;16:00
Mac Address of ESP32;Device UUID;Event Status;Event Time
```









Features

Quality of Service
Persistent Session and Queuing
Retained Messages
Last Will and Testament
Keep Alive and Client Take-Over
Other Protocols

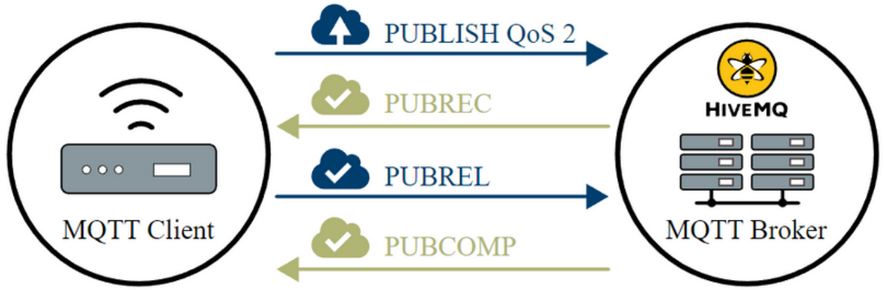
Quality of Service (QOS)

An agreement between the sender of a message and the receiver of a message that defines the guarantee of delivery for a specific message.

Quality of Service level 0: delivery at most once

Quality of Service level 1: delivery at least once

Quality of Service level 2: delivery exactly once



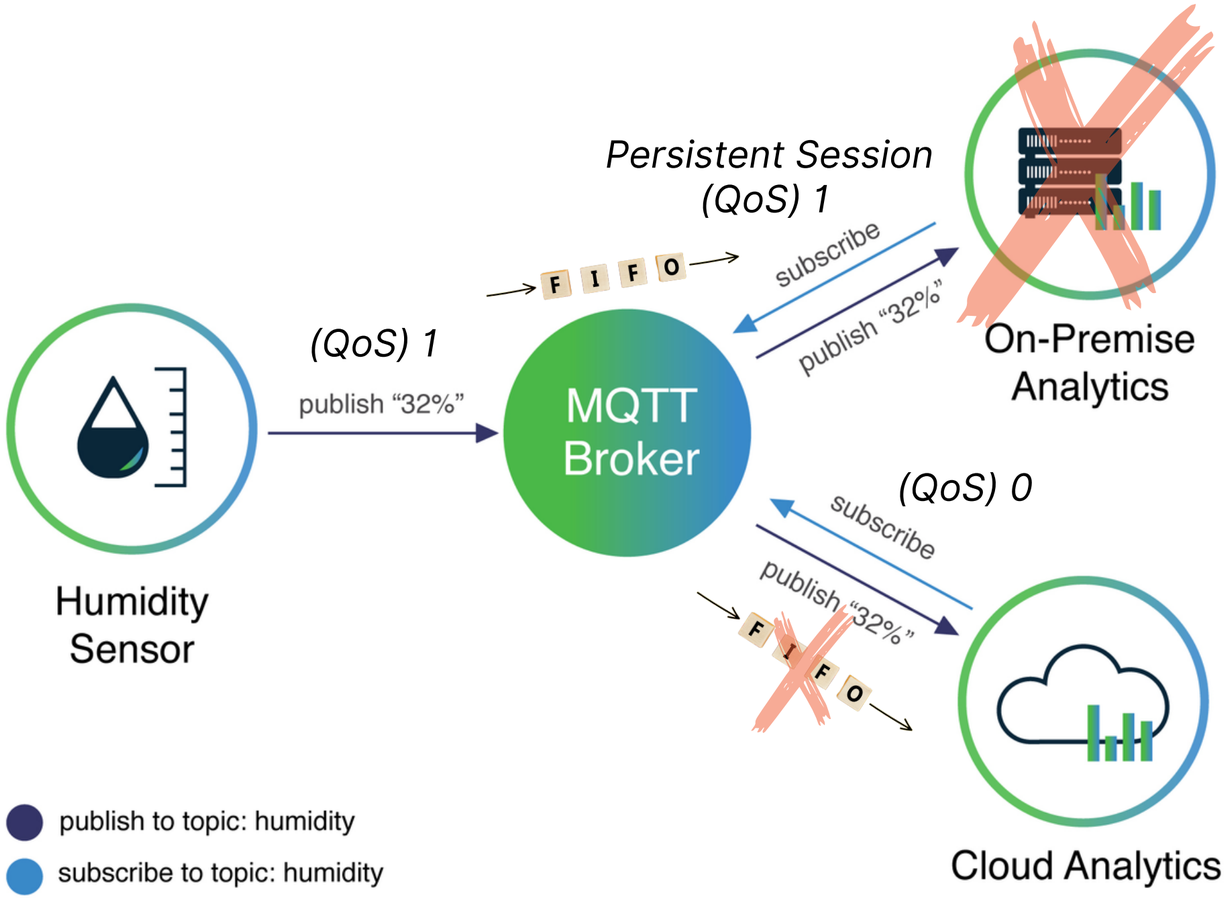
All messages sent with QoS 1 and 2 are **queued for offline clients until the client is available again.** However, this queuing is only possible if the client has a **persistent session.**

Persistent Session and Queuing Messages

In a persistent session

Existence of a session (even if there are no subscriptions).
All the subscriptions of the client.
All messages in a Quality of Service (QoS) 1 or 2 flow that the client has not yet confirmed.
All new QoS 1 or 2 messages that the client missed while offline.
All QoS 2 messages received from the client that are not yet completely acknowledged.

Persistent Session and Queuing Messages



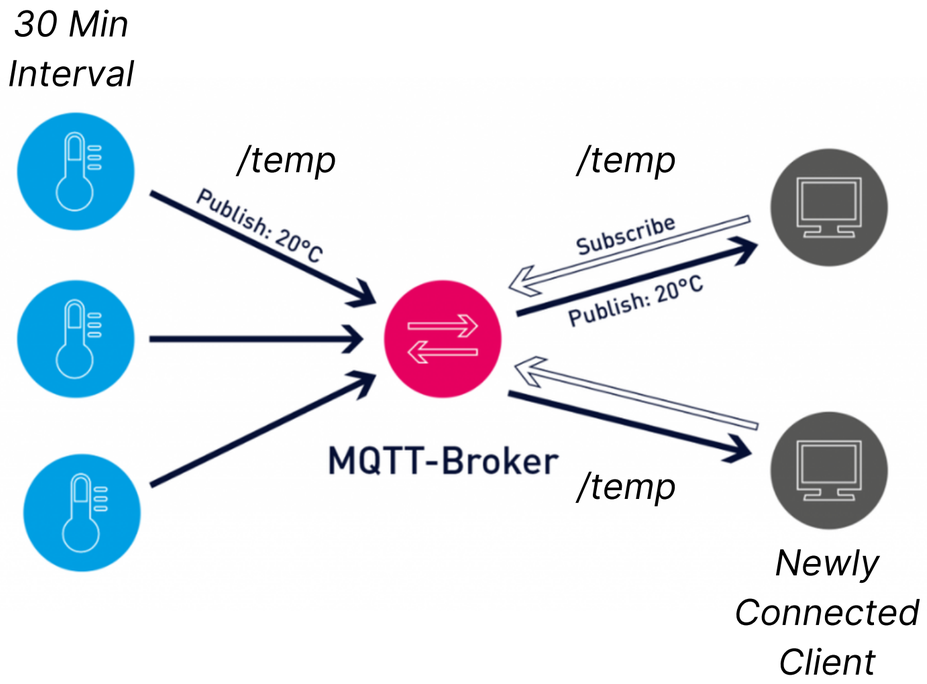
borda
IoT for Healthcare

Retained Messages

Last known good value.

A retained message is a normal MQTT message with the **retained flag set to true.** The broker stores the last retained message and the corresponding QoS for that topic.

The broker stores only one retained message per topic.



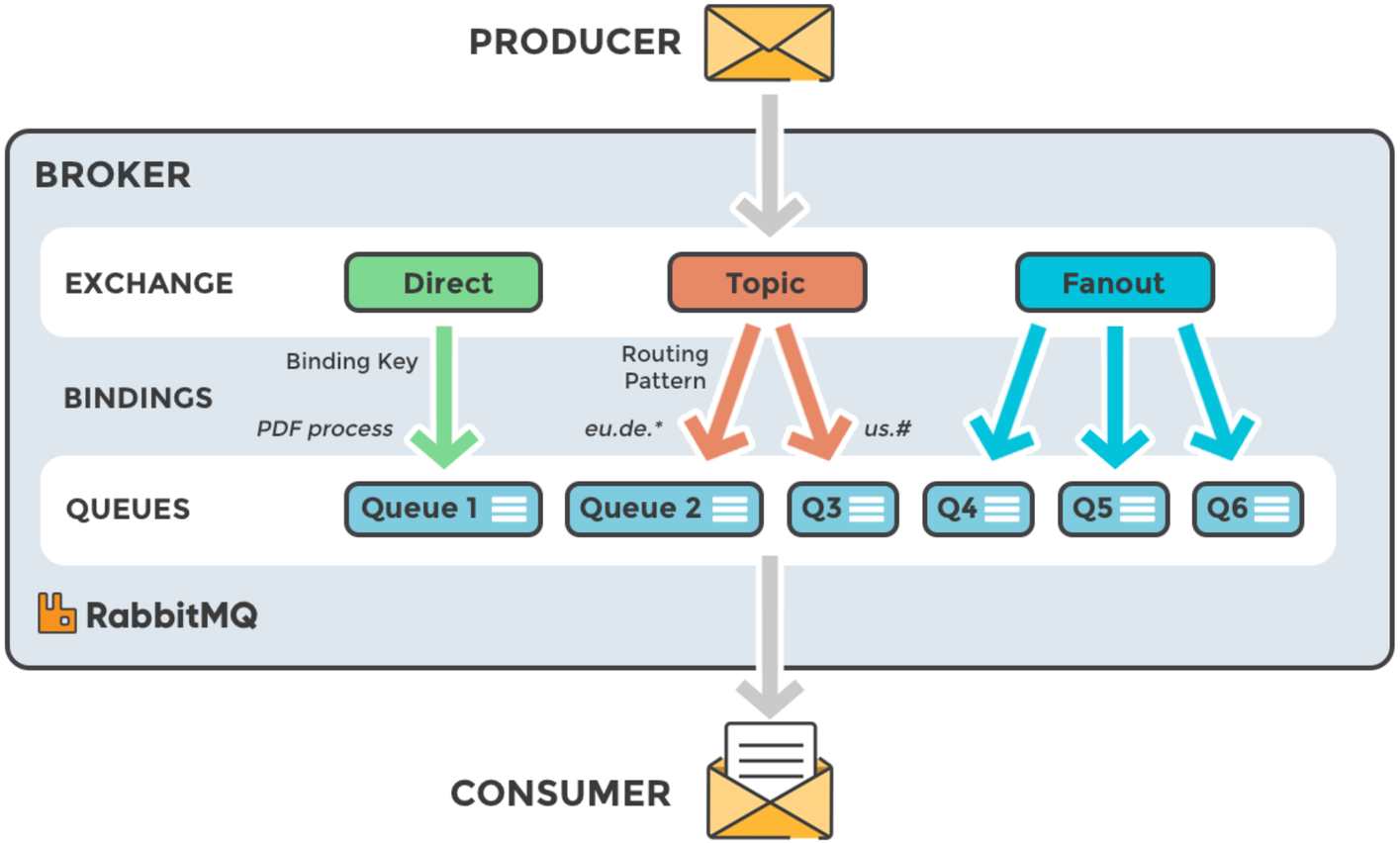
Retained messages help newly-subscribed clients **get a status update immediately after they subscribe to a topic.**

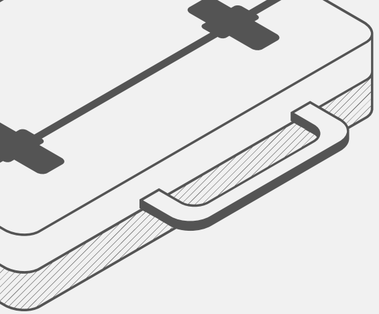
The retained message eliminates the wait for the publishing clients to send the next update.

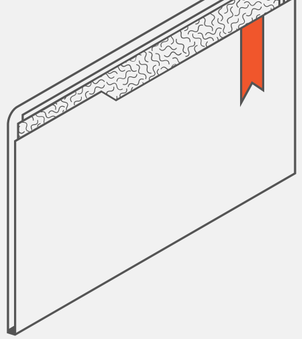
borda
IoT for Healthcare

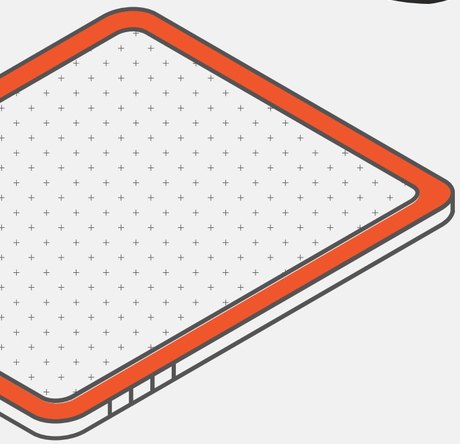
Other Protocols

MQTT 3.1.1 vs MQTT 5 vs AMQP





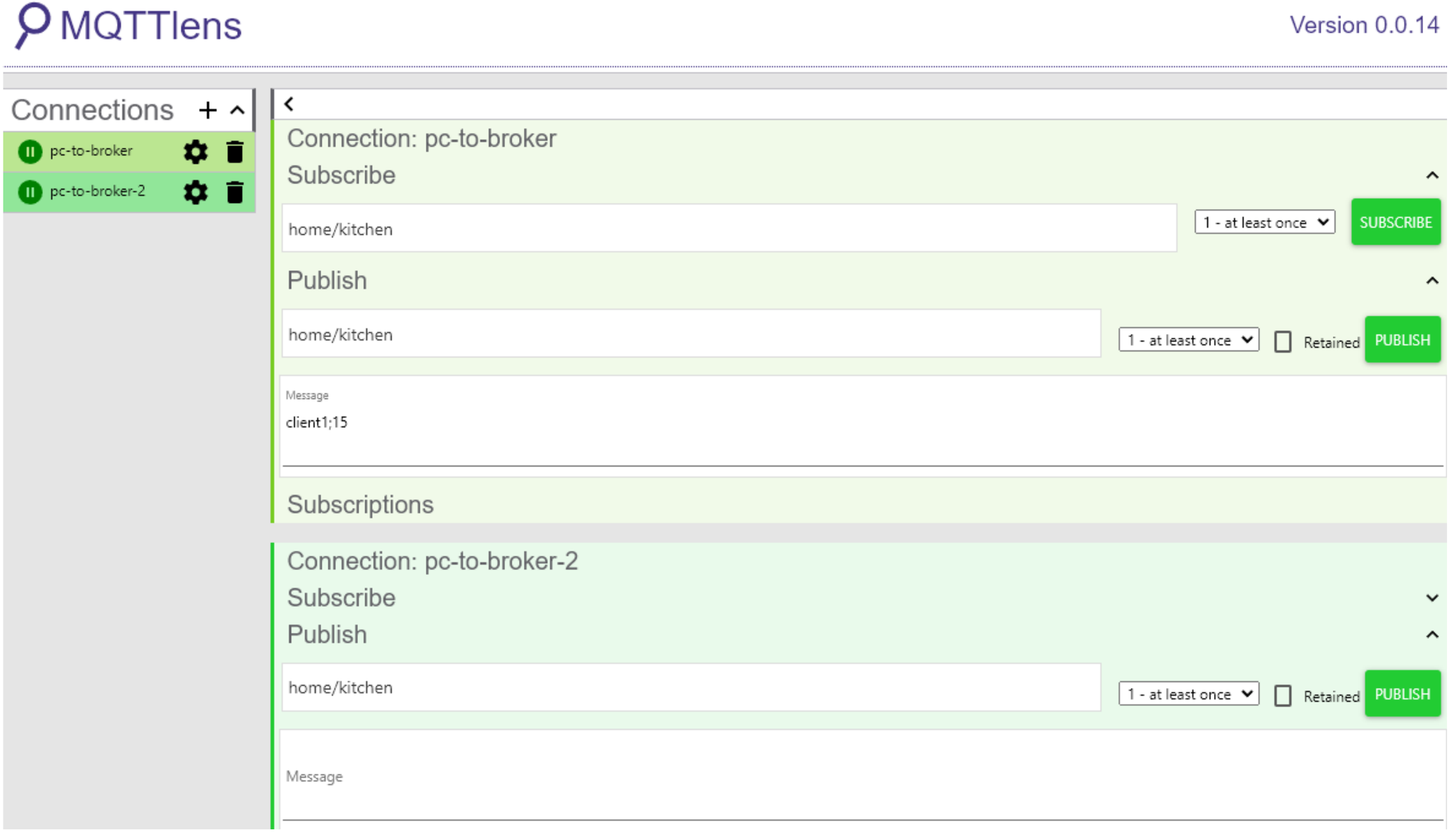




Demo

RabbitMQ Broker
MQTT Lens
ESP to ESP Communication
Further Readings

borda
IoT for Healthcare


MQTTlens
Version 0.0.14
Connections
pc-to-broker
pc-to-broker-2
Connection: pc-to-broker
Subscribe
home/kitchen
1 - at least once
SUBSCRIBE
Publish
home/kitchen
1 - at least once
Retained
PUBLISH
Message
client1;15
Subscriptions
Connection: pc-to-broker-2
Subscribe
Publish
home/kitchen
1 - at least once
Retained
PUBLISH
Message

borda
IoT for Healthcare

RabbitMQ

Features | Get Started | Support | Community | Docs | Blog

Quorum queues

A webinar on high availability
and data safety in messaging

Learn more

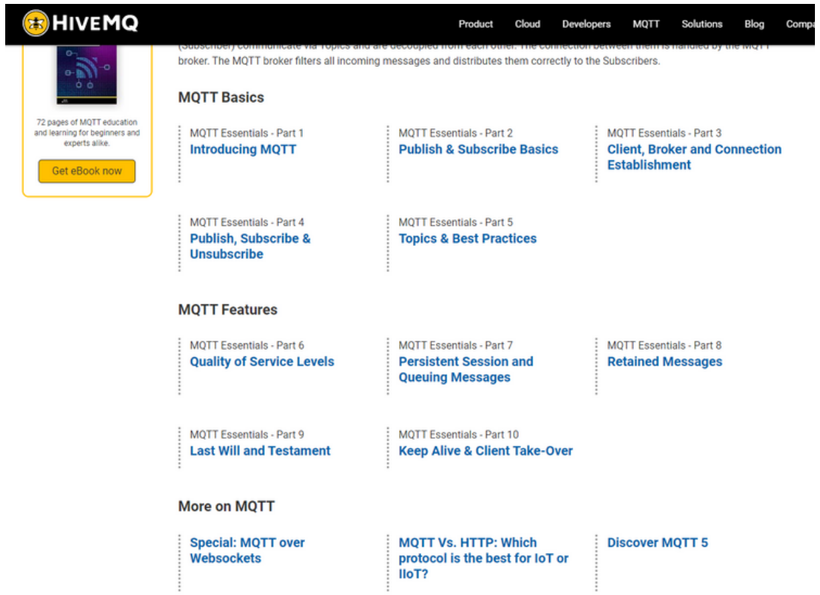
RabbitMQ is the most widely deployed open source message broker.

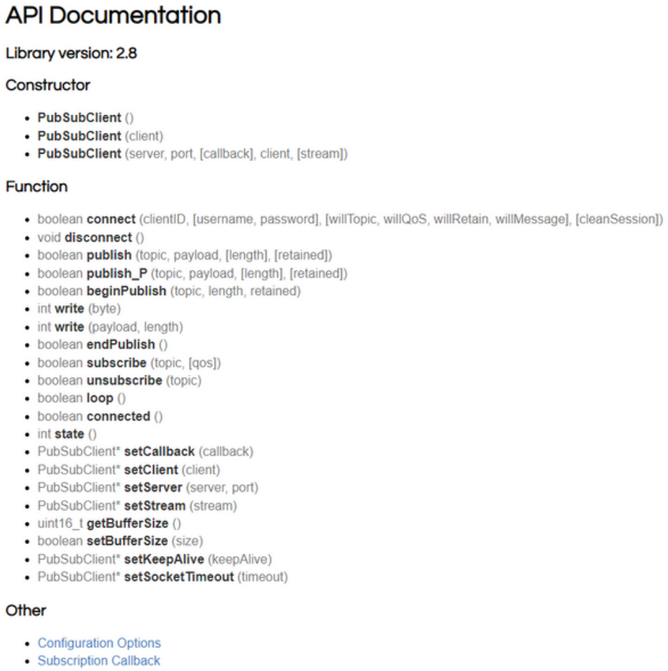
Updates

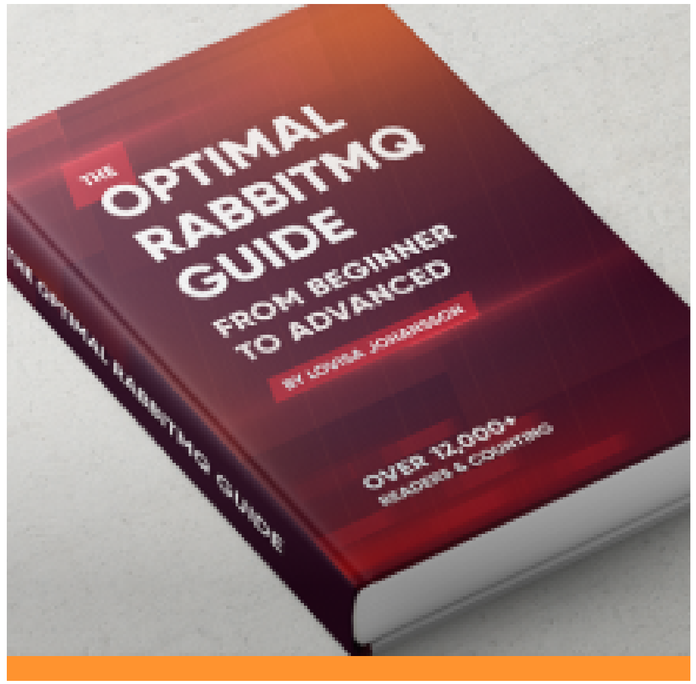
1. RabbitMQ 3.10.5 — 06 Jun 2022
2. RabbitMQ 3.9.20 — 06 Jun 2022

borda
IoT for Healthcare

Further Readings







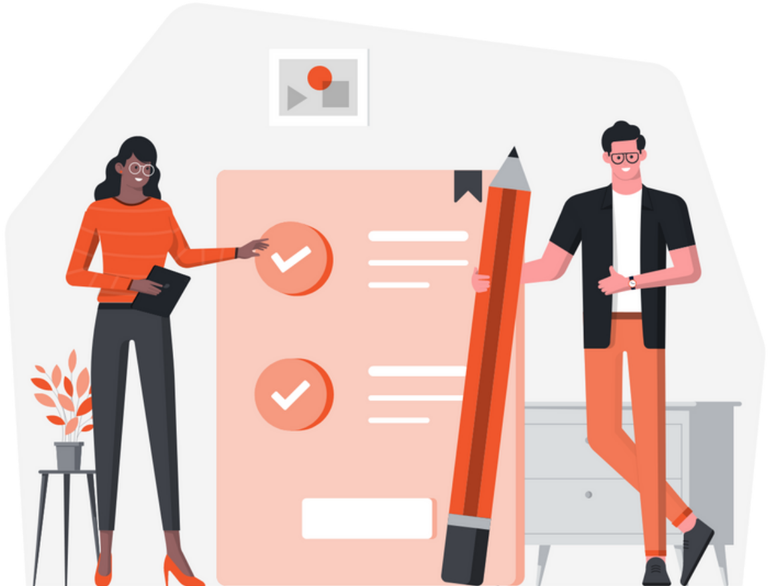
[HiveMQ MQTT Essentials]

[PubSubClient API Docs]

[RabbitMQ Book]

To Do List / HW

1. What are the **main limitations** of the MQTT?
2. What the other 8 billion devices use **instead**/or with MQTT?
3. Can you explaing the **publish/subscribe** pattern to the five year old? (ELI5)
4. Why **IBM** made the MQTT **Royalty-Free**?

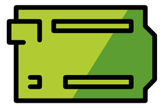


borda
IoT for Healthcare

To Do List / HW

JSON Goes Here — 167 B (164 B)

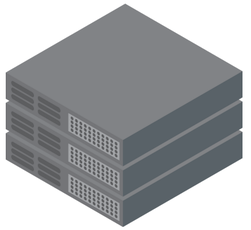
```
{
   "temp":"25.6,25.6,23.6,22.6,21.6,23.6,24.6,25.6,26.6,27.6"
   "mac_id":"00:00:5e:00:53:af",
   "humid":"50.6,51.6,52.6,53.6,54.6,55.6,56.6,57.6,58.6,59.6"
}
```



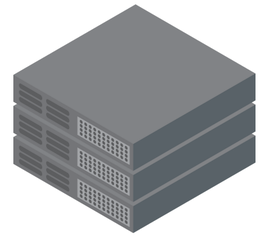
1000 Sensor Publisher
10 Min interval

167 Payload Size
+ 4 Byte Overhead

Publisher



Subscriber



Subscriber

Sensors include every 1 minutes Temperature and Humidity data and sends them with 10 minutes intervals.

1 . What would be the maximum required bandwidth on the broker?

2. What would be the minimum bandwidth on the broker at any time?

3. Can you decrease the required bandwidth?

https://www.javainuse.com/bytesizejson

{
"temp":"25.6,25.6,23.6,22.6,21.6,23.6,24.6,25.6,26.6,27.6"
"mac_id":"00:00:5e:00:53:af",
"humid":"50.6,51.6,52.6,53.6,54.6,55.6,56.6,57.6,58.6,59.6"
}