Back to Basics

An Introduction to MQTT

Mary Grygleski
Senior Developer Advocate
at HiveMQ

Speaker

Mary Grygleski
Senior Developer Advocate at HiveMQ

@mgrygles
https://www.linkedin.com/in/mary-grygleski/
https://www.twitch.tv/mgrygles
https://discord.gg/RMU4Juw

Based out of Chicago, Mary is a Java Champion and President and Executive Board Member of the Chicago Java Users Group (CJUG). She is also the co-organizer for several meetup groups such as, the *Data, Cloud and AI In Chicago*, *Chicago Cloud*, and *IBM Cloud Chicago*.

She has extensive experience in product and application design, development, integration, and deployment experience, and specializes in Reactive Java, Open Source, and cloud-enabled distributed systems.

AGENDA

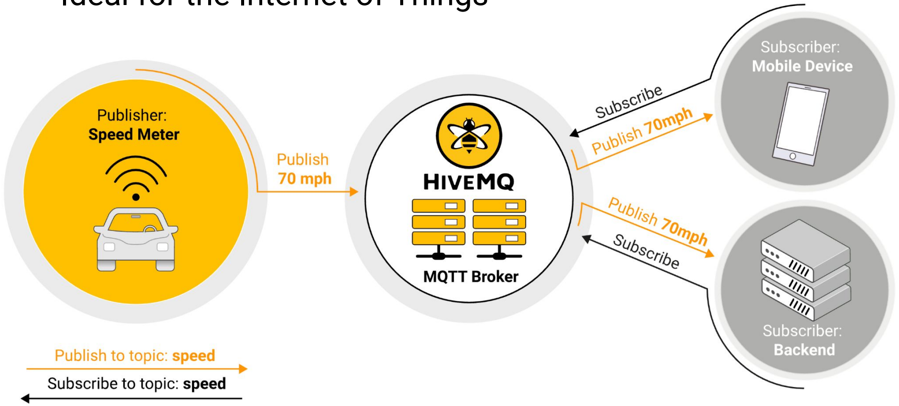
- ❑ **The IoT Stack**
- ❑ **What is MQTT?**
  - ❑ A bit of a history
  - ❑ OASIS open standard, community-driven protocol
  - ❑ MQTT 3.1.1 Features
  - ❑ MQTT 5.0 Features
- ❑ **Use Cases for MQTT**
- ❑ **Alternative Protocols for IoT**
- ❑ **Summary - Why MQTT is the top choice**
- ❑ **Demo**

From 30,000 Feet: The IoT Stack

Solutions	**Consumer** (Home, Lifestyle, Mobility, Entertainment…etc.)	**Enterprise** (ERP, Marketing, Sales, Business Ops…etc.)	**Industrial** (Manufacturing, Automotive, Logistics, Construction…etc.)
Cognitive Platform	Interactions (spoken, gestures), AI, Voice Recognition…etc.		
Analytics Platform	Services (Geospatial, Recommendations), Reports, Events, Machine Learning, Stream Processing…etc.		
Core Platform	IoT Messaging Middleware, Protocol Gateway, Data Aggregation, Data Storage/Filter		
Communication Protocols	MQTT, CoAP, AMQP, DDS, XMPP, IPv4, IPv6, 6LoWPan,Bluetooth, GSM, Modbus, BACnet		
Devices	Devices & Components: Smart Devices, Sensors & Actuators, Embedded Devices…etc.		

What is MQTT?

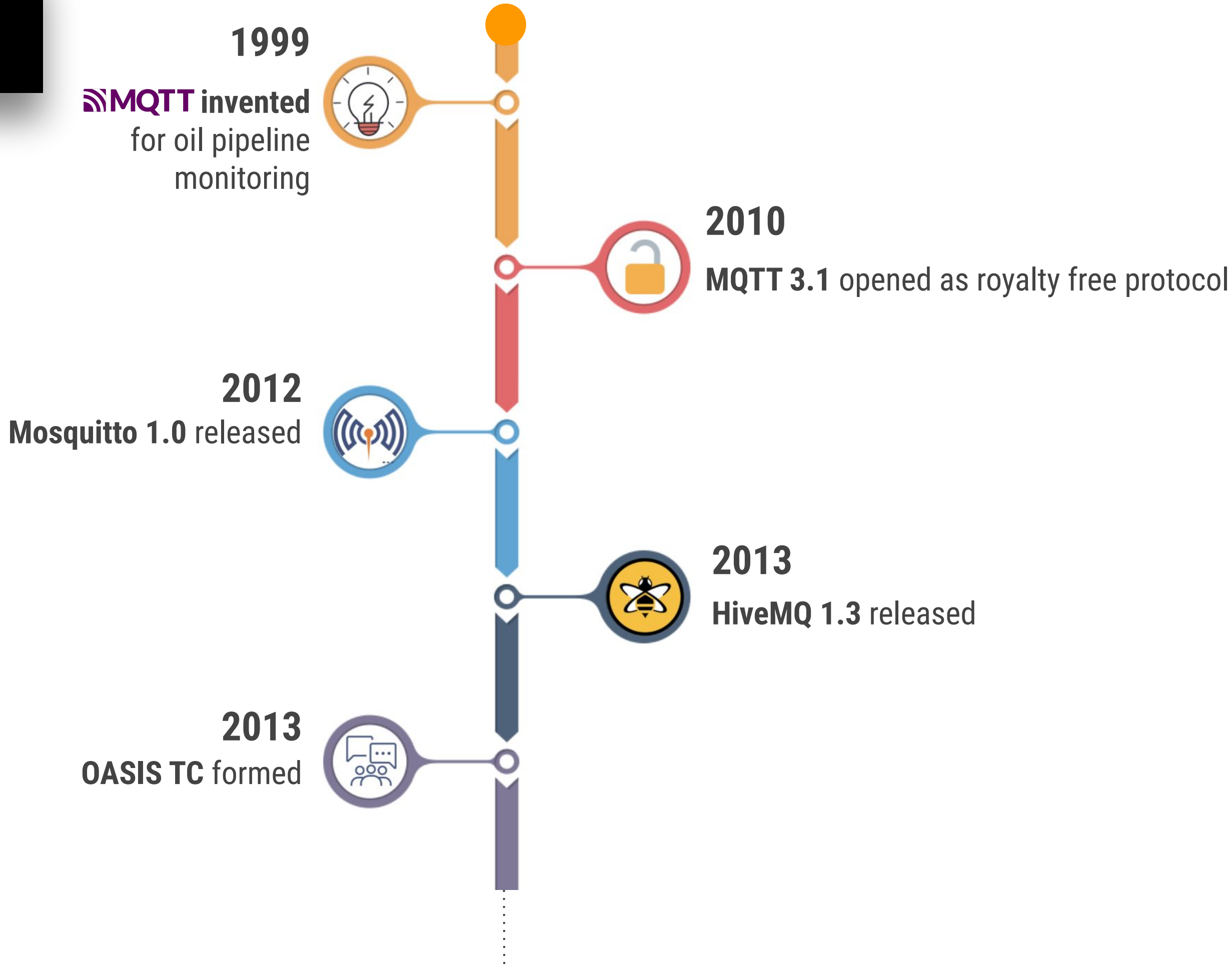
- A standard binary publish-subscribe messaging protocol designed for fast and reliable data transport between devices especially under very constrained conditions
- Constraints include unreliable network connectivity, limited bandwidth, limited battery power, and so on
- Built on top of TCP/IP
- Ideal for the Internet of Things



A Brief History of MQTT

- Invented in 1999 by Andy Stanford-Clark at IBM and Arlen Nipper at what was then Arcom - and now Cirrus Link
- Prompted by the need to design a protocol that could handle a very limited operating environment that can afford only minimal battery loss and minimal bandwidth to connect with oil pipelines via satellite

History…



1999

MQTT invented for oil pipeline monitoring

2010

MQTT 3.1 opened as royalty free protocol

2012

Mosquitto 1.0 released

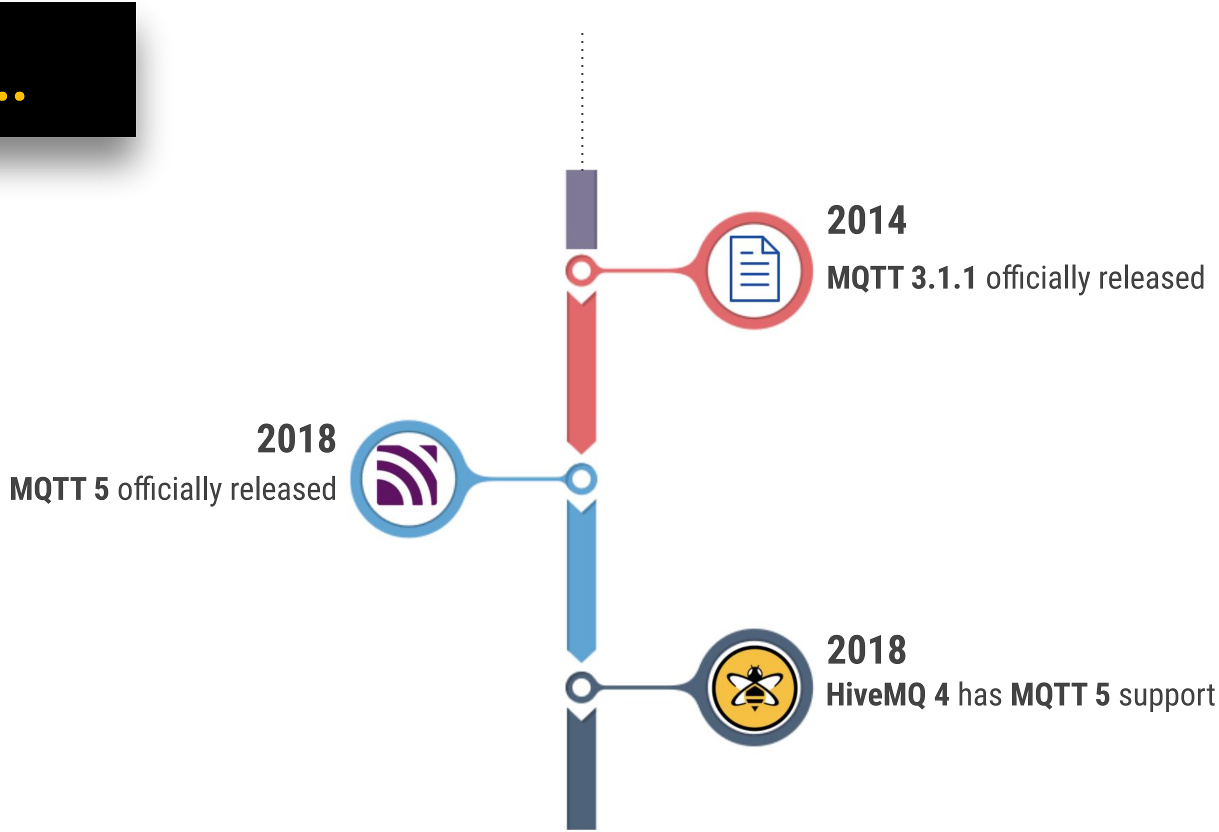
2013

HiveMQ 1.3 released

2013

OASIS TC formed

History...



2014

MQTT 3.1.1 officially released

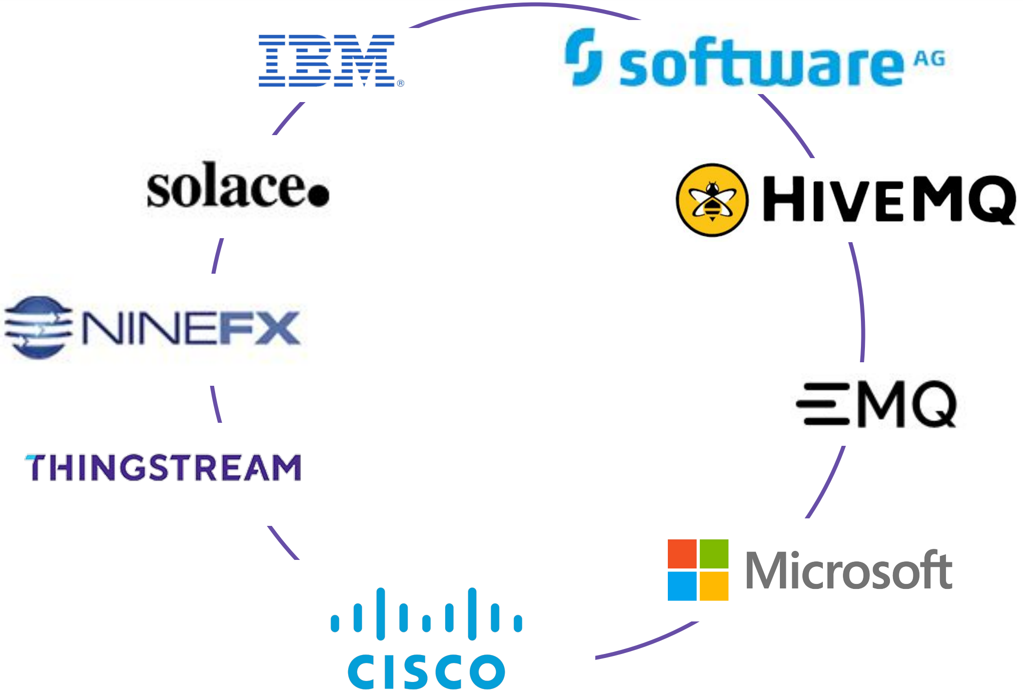
2018

MQTT 5 officially released

2018

HiveMQ 4 has **MQTT 5** support

THE MQTT Technical Committee



IBM

software AG

HIVEMQ

EMQ

Microsoft

CISCO

THINGSTREAM

NINEFX

solace

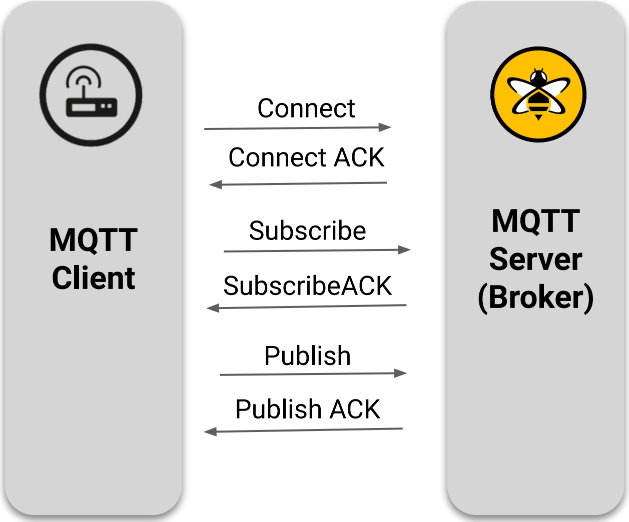
MQTT Overview

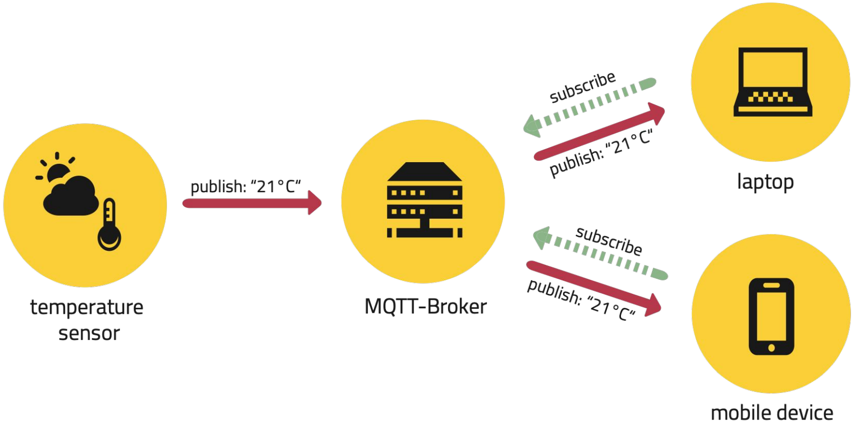
- IoT Messaging Protocol
- 3 QoS Levels
- Retained Messages
- Stateful - persistent sessions
- Binary with minimal overhead

Basic Features of MQTT 3.1.1

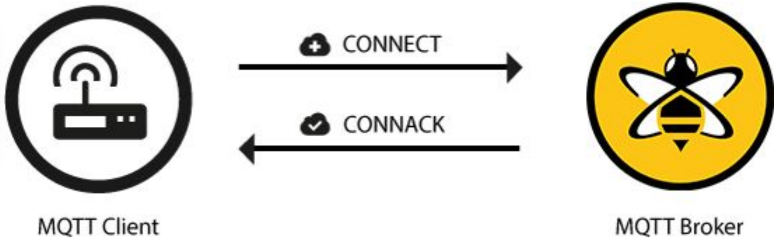
HIVEMQ

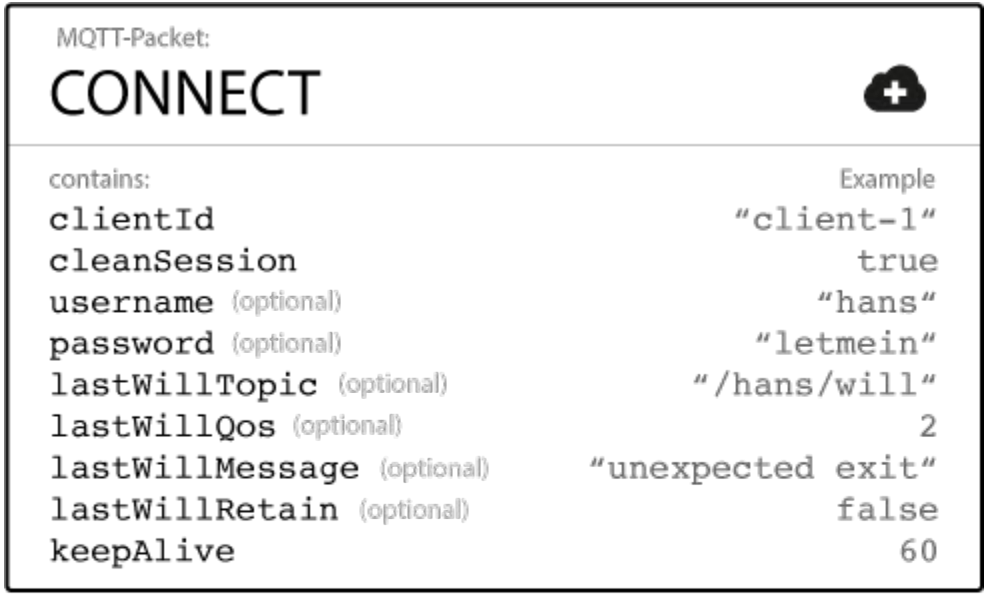
Publish / Subscribe





CONNECT / CONACK

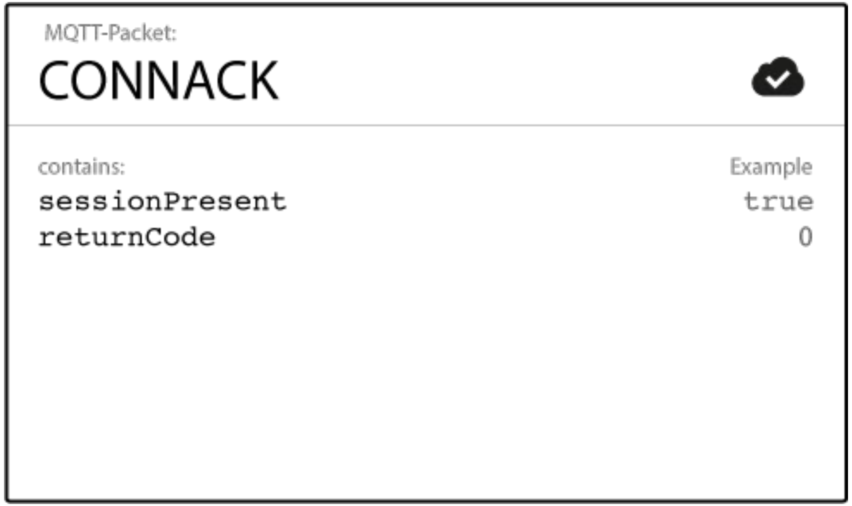




MQTT-Packet:

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



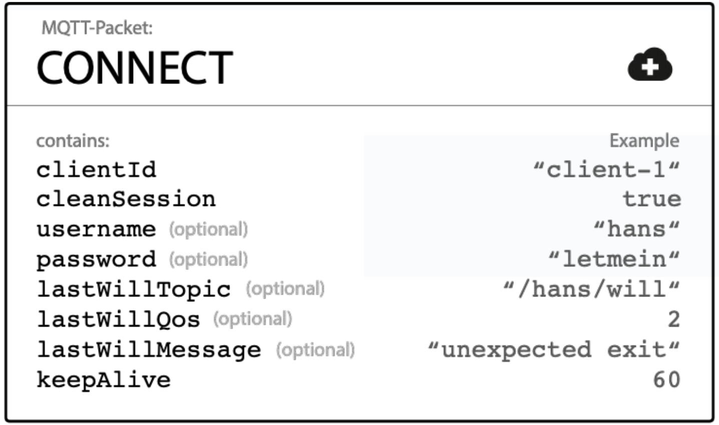
MQTT-Packet:

CONNACK

contains:	Example
sessionPresent	true
returnCode	0

WILL

- Client defines Will (LWT)
- Broker sends this message if this client dies
- It is a real Push
- Useful to implement on / off mechanism in a safe way
- message when Subscribing to the topic

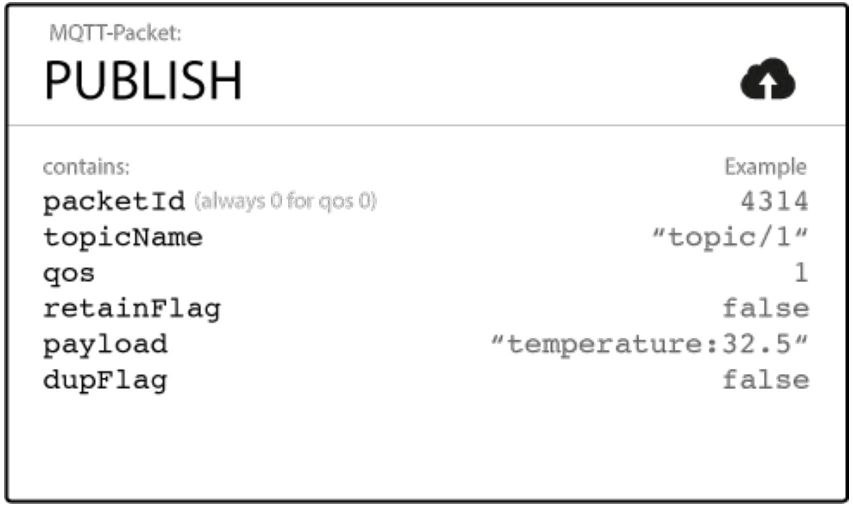


MQTT-Packet:

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“
`keepAlive`	60

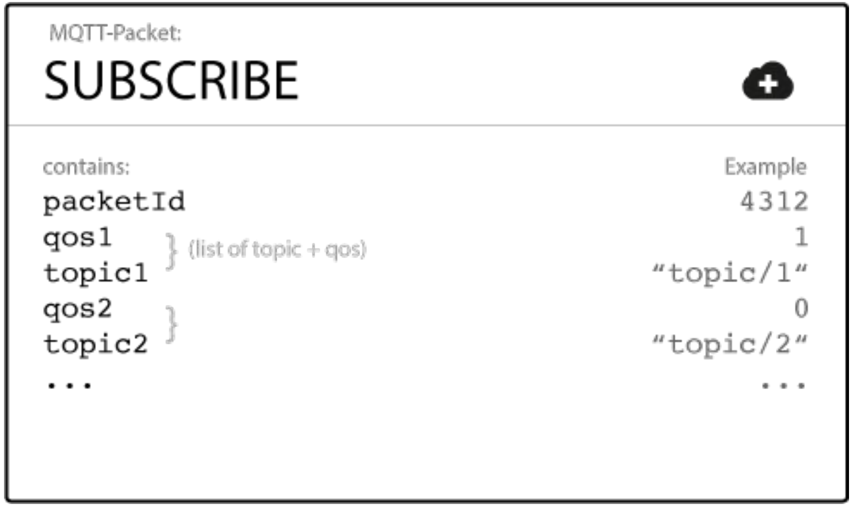
Publish / Subscribe



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



MQTT-Packet:

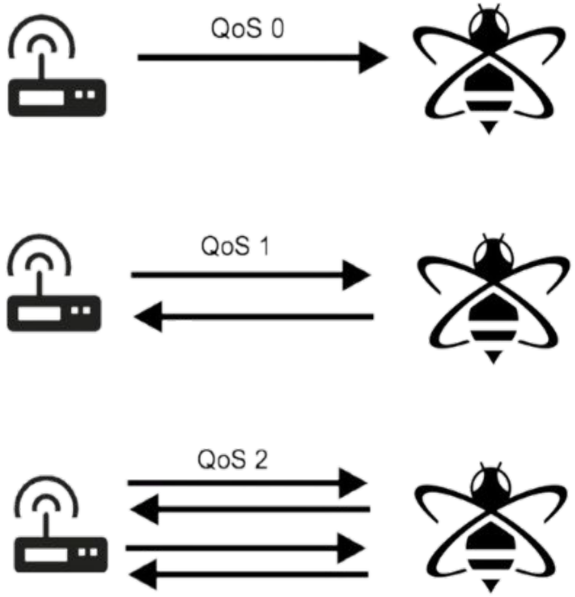
SUBSCRIBE

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

Retained Message

- Last Known “Good Value”
- Last message will be stored on broker side
- Client decides if a message is retained or not
- Future Clients get the retained message when Subscribing to the topic

Quality of Services



QoS 0 | At most once delivery

QoS 1 | At least once delivery

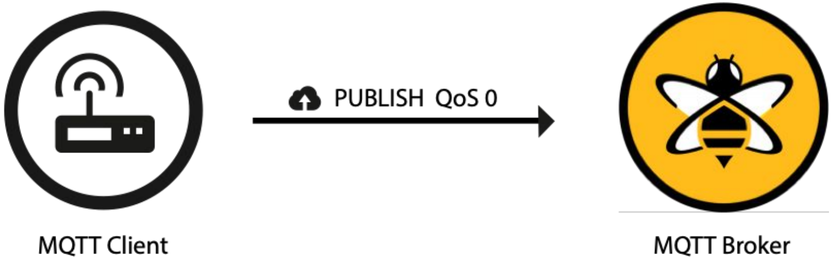
QoS 2 | Exactly once delivery

Quality of Services 0

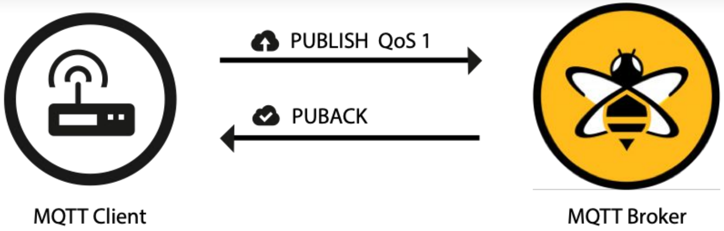
MQTT-Packet:

PUBLISH

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



Quality of Services 1



MQTT-Packet:

PUBLISH

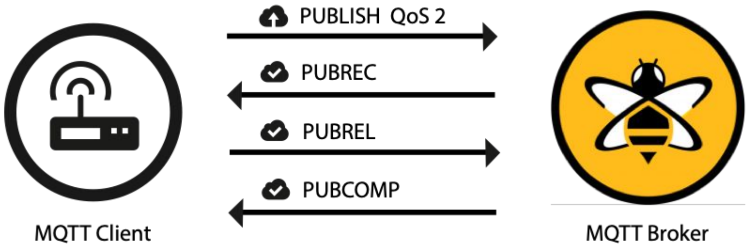
contains:	Example
`packetId` (always 0 for qos 0)	`4314`
`topicName`	`“topic/1“`
`qos`	`1`
`retainFlag`	`false`
`payload`	`“temperature:32.5“`
`dupFlag`	`false`

MQTT-Packet:

PUBACK

contains:	Example
`packetId`	`4314`

Quality of Services 2



MQTT-Packet:

PUBLISH

contains:	Example
packetId (always 0 for qos 0)	4314
topicName	"topic/1"
qos	2
retainFlag	false
payload	"temperature:32.5"
dupFlag	false

MQTT-Packet:

PUBREC

contains:	Example
packetId	4314

MQTT-Packet:

PUBREL

contains:	Example
packetId	4314

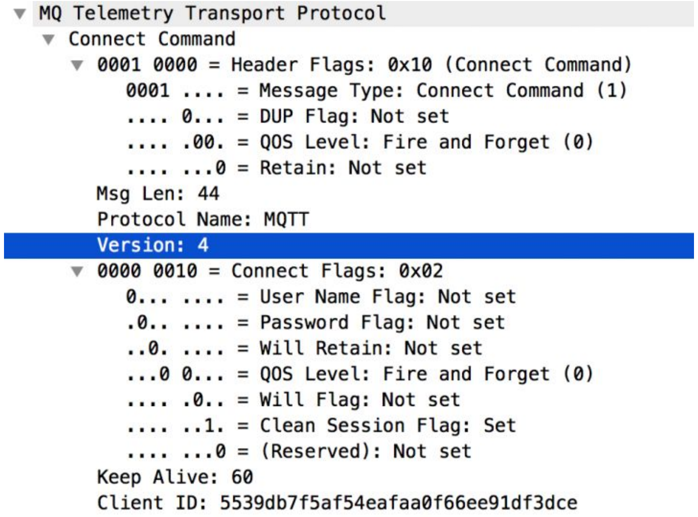
MQTT-Packet:

PUBCOMP

contains:	Example
packetId	4314

But, where is MQTT 4 ?

Hint: let’s look at the CONNECT message packet detail for MQTT 3.1.1:



```
▼ MQ Telemetry Transport Protocol
  ▼ Connect Command
    ▼ 0001 0000 = Header Flags: 0x10 (Connect Command)
        0001 .... = Message Type: Connect Command (1)
        .... 0... = DUP Flag: Not set
        .... .00. = QOS Level: Fire and Forget (0)
        .... ...0 = Retain: Not set
      Msg Len: 44
      Protocol Name: MQTT
      Version: 4
    ▼ 0000 0010 = Connect Flags: 0x02
        0... .... = User Name Flag: Not set
        .0.. .... = Password Flag: Not set
        ..0. .... = Will Retain: Not set
        ...0 0... = QOS Level: Fire and Forget (0)
        .... .0.. = Will Flag: Not set
        .... ..1. = Clean Session Flag: Set
        .... ...0 = (Reserved): Not set
      Keep Alive: 60
      Client ID: 5539db7f5af54eafaa0f66ee91df3dce
```

MQTT 5

MQTT 5 - Overview

- Successor of MQTT 3.1.1
- Non-backward compatible
- First public release in January 2018, official release in March 2019
- Many new features
- Clarifications of the 3.1.1 specification

MQTT 5 - Goals

- Enhancements for scalability and large scale systems
- Improved error reporting
- Formalize common patterns including capability discovery and request response
- Extensibility mechanisms including user properties
- Performance improvements and support for small clients

NEW FEATURES

Session & Message Expiry

- Session Expiry is an optional part of the CONNECT message
- Session Expiry Interval in Seconds
- Broker expires session after the given interval as soon as the client disconnects
- Publication Expiry interval is an optional part of a PUBLISH message
- Applies to online and queued messages

User Properties

- User Defined Metadata Headers
- Can be part of most MQTT packets (CON, PUB, SUB)
- UTF-8 encoded Strings
- An unlimited number of user properties can be added

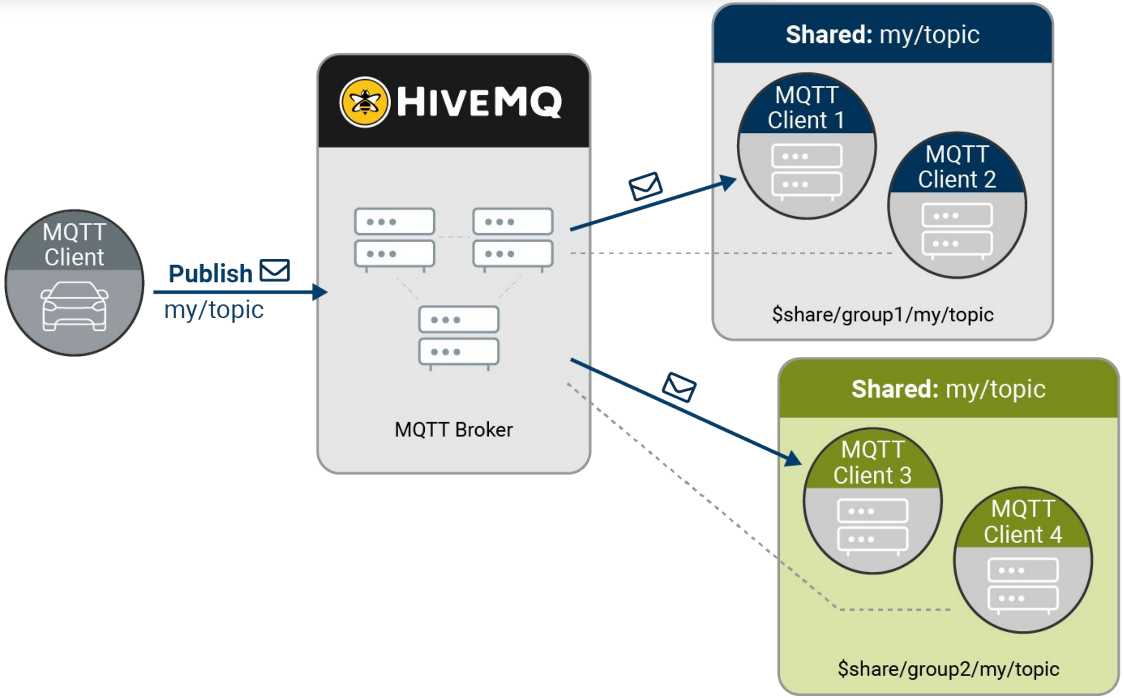
Shared Subscriptions

Special Syntax: $share/{ID}/my/topic

- Useful for scaling out backend subscribers
- Client Load Balancing. Multiple clients share the same subscription
- Also supported by HiveMQ for MQTT 3.1 and MQTT 3.1.1
- Up-/Downscaling of clients at runtime possible. Perfect for cloud native scenarios (Kubernetes, …)
- Optional feature, not supported by all vendors*

** HiveMQ fully supports all optional features, including this feature*

Shared Subscriptions

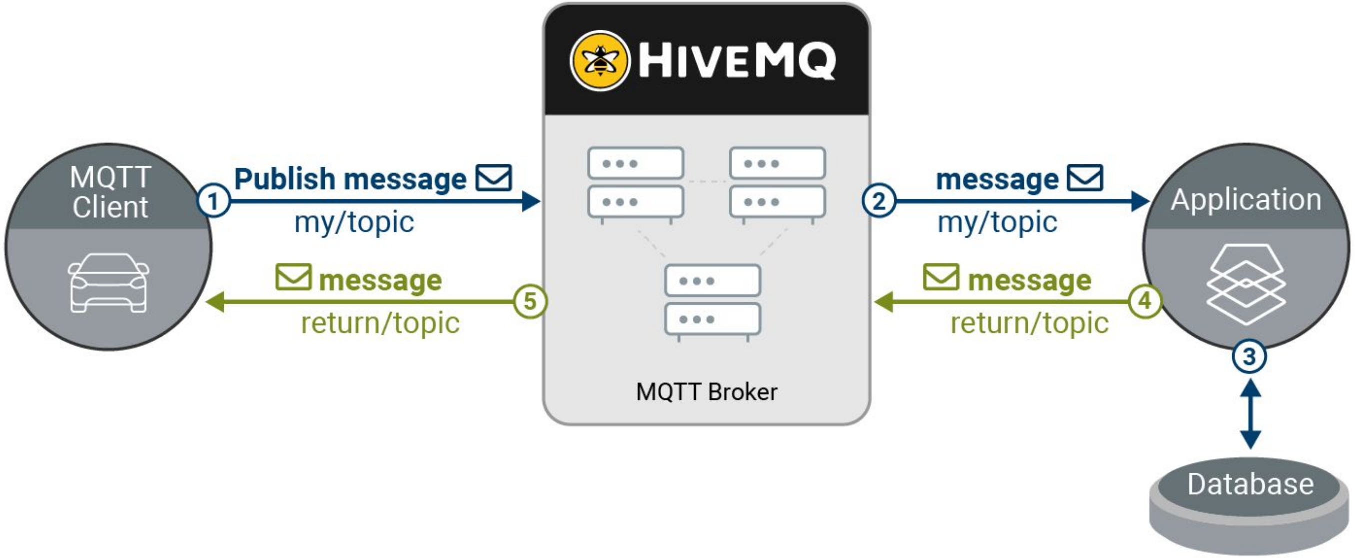


Request / Response

Pattern for “business ACKs”

- The MQTT request-response pattern is not the same as the request-response of synchronous, client-server based protocols like HTTP.
- Request as well as responses are at least Topics and can have more than one or no subscriber in MQTT.
- The Client must subscribe to a response topic prior to sending data.
- “Request Response Information” place for response topic
- “Correlation Data” header for correlation of the request and its response

Request / Response



Lightweight and Bandwidth Efficient

- Every message works as a discrete chunk of data, opaque to the broker
- MQTT Control packet structure:
  - Fixed header, Variable header, Payload
- Protocol headers are small in size:
  - 2 byte fixed header
- up to 12 bytes of additional variable header (variable size and present only when needed)

Data Agnostic

- Supports all kinds of data:
    - images
    - text in any encoding format
    - encrypted data
    - binary data

Continuous Session Awareness

- Persistent sessions
  - Broker store messages when offline
  - QoS level 1
  - Retained messages
  - Normal message with "retain" flag will be stored and sent to new subscribers to its topic
  - Last will and testaments
  - Client can specify a message to send in case it disconnects ungracefully
- Very useful in IoT especially over unreliable networks

MQTT 5

- Introduction of semantic metadata like user properties, payload indicators, or content type descriptors
- Request-response pattern
- Shared subscriptions
- Negative acknowledgments
- Message and session expiry per client
- More…

Use Cases for MQTT

- IoT
- Industrial IoT (IIoT)
- Industry 4.0
- Industry verticals:
  - Automotive
  - Logistics
  - Manufacturing
  - Energy
- Consumers:
  - Smart Home
  - Lifestyle

Alternative Protocols to MQTT

- HyperText Transport Protocol (HTTP)
- Constrained Application Protocol (CoAP)
- Advanced Messaging Queueing Protocol (AMQP)
- Object linking & embedding for Process Control - Unified Architecture (OPC-UA)
- Data Distribution Service (DDS)
- Extensible Messaging and Presence Protocol (XMPP)

Integration with Other Frameworks

- Streaming platforms: Apache Kafka
- Other MQTT Brokers: Mosquitto
- Runtimes - SpringBoot

Summary

- Simplicity - Pub/Sub - Asynchronous processing - Loosely coupled
- Lightweight
- Operating in a constrained environment
- Unreliable, high latency network
- Limited battery and other resources
- Ideal protocol for IoT use cases (other protocols such as HTTP would be too heavy)

Demo

Resources

Get Started with MQTT

MQTT Essentials Series

MQTT at OASIS

Evaluate HiveMQ Broker

Try HiveMQ Cloud

ANY QUESTIONS?

Reach out to community.hivemq.com

THANK YOU

Contact Details

Mary Grygleski

mary.grygleski@hivemq.com

https://www.linkedin.com/in/mary-grygleski/

@mgrygles

https://discord.gg/RMU4juw

https://www.twitch.tv/mgrygles