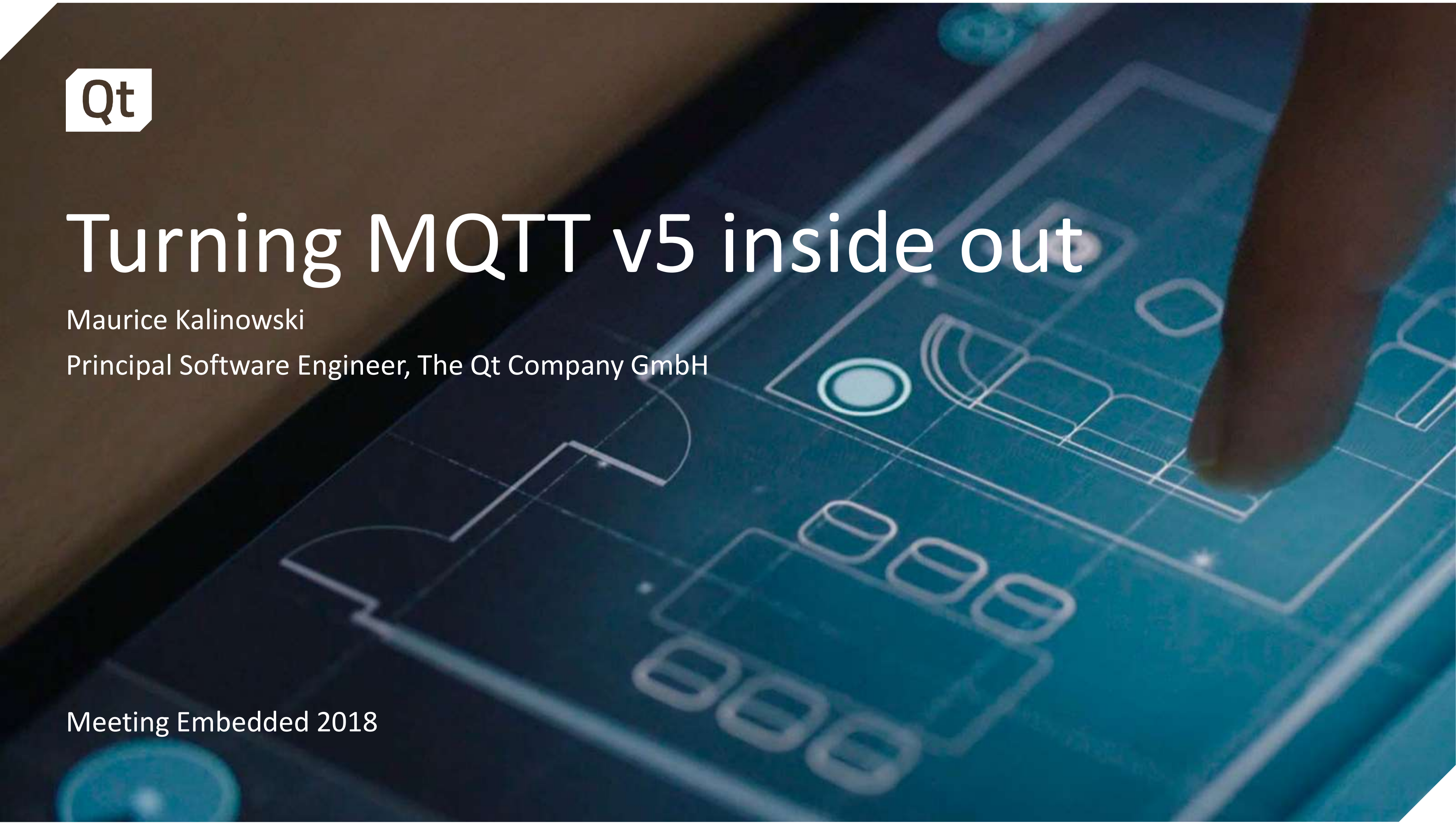


Turning MQTT v5 inside out

Maurice Kalinowski

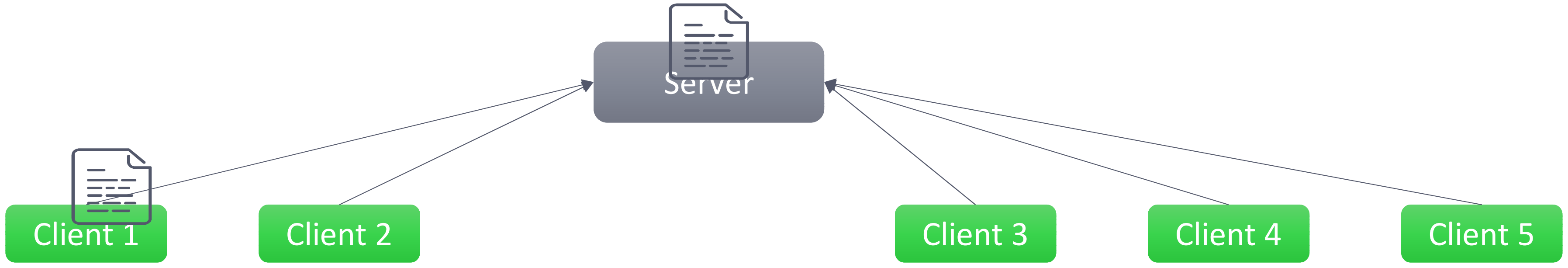
Principal Software Engineer, The Qt Company GmbH

Meeting Embedded 2018

About the Speaker

- Maurice Kalinowski
  - The Qt Company
  - Maintainer for Qt MQTT
  - …

Publish and Subscribe



clientx.connectToHost(…)

Client3.subscribe(“Topics/t1”)

Client1.publish(“Topics/t1”, “someData”)

Client4.subscribe(“Topics/t2”)

Client5.subscribe(“Topics/#”)

onMessageReceived(…) {…}

Why MQTT?

- Open Standard
- Freedom of choice
  - Many implementations exist
  - Different programming languages
  - Different licenses
  - …
- Interoperability
  - Integration options to cloud solutions:
    - Amazon
    - Azure
    - …
  - Integration options to other M2M protocols
    - OpcUA decided for MQTT for Pub/Sub in the standard

Why MQTT?

- MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple and designed 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. The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bidirectional connections. Its features include:
  - Use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications.
  - A messaging transport that is agnostic to the content of the payload.
  - Three qualities of service for message delivery
    - At most once
    - At least once
    - Exactly once
  - A small transport overhead and protocol exchanges minimized to reduce network traffic.
  - A mechanism to notify interested parties when an abnormal disconnection occurs

Why MQTT?

- Open Standard by OASIS
  - As an M2M/Internet of Things (IoT) connectivity protocol, MQTT is designed to support messaging transport from remote locations/devices involving small code footprints (e.g., 8-bit, 256KB ram controllers), low power, low bandwidth, high-cost connections, high latency, variable availability, and negotiated delivery guarantees.
- Quoting MQTT.org
  - It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium.

MQTT.ORG

OASIS
Open standards. Open source.

Lightweight?

Packet Layout

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Connect	Length		“MQTT”				Version	Flags	Keep Alive		Payload	… e.g. username		
Subscribe	Length		ID		Topic Length		“a/b”			QoS				
Publish	Length		Topic Length		“a/b”			Packet ID		Payload / Message Content				

Only QoS 1/2

7	Publish Statement
6	
5	
4	
3	DUP
2	QoS
1	Level
0	Retain

- Minimal overhead per command
  - Some commands merge options into command statement
- Length types are “Variable byte Integers” with byte size 1-4

Available commands

Command	Value	Description	Direction
CONNECT	0x10	Request Connection	C->S
CONNACK	0x20	Connection request accepted	S->C
PUBLISH	0x30	Send/Receive message	C->S
PUBACK	0x40	Message has been received and handled by server (QoS1)	S->C
PUBREC	0x50	Message has been received (QoS2)	S->C
PUBREL	0x60	Message can be released (QoS2)	C->S
PUBCOMP	0x70	Message handling completed (QoS2)	S->C
SUBSCRIBE	0x80	Subscribe to one or more topics	C->S
SUBACK	0x90	Subscription request has been accepted	S->C
UNSUBSCRIBE	0xA0	Remove subscription	C->S
UNSUBACK	0xB0	Subscription removal done	S->C
PINGREQ	0xC0	Ping	C->S
PINGRESP	0xD0	Pong	S->C
DISCONNECT	0xE0	Request Clean Disconnect	C->S

MQTT v5

- Protocol level 5

MQTT protocol level 5.0

Payload format and content type

Flow Control

Message Expiry

Maximum Packet Size

Session Expiry

Will delay

Shared Subscriptions

Topic Alias

Subscription Options

(User) Properties

Subscription IDs

Reason Codes

Enhanced Authentication

Server reference

Request / Response

Server disconnect

Reason Strings

Assigned Client ID

Server Keep Alive

Server feature / capability management

MQTT protocol level 5.0

Payload format and content type

Message Expiry

Flow Control

Maximum Packet Size

Session Expiry

Will delay

Shared Subscriptions

Topic Alias

Subscription Options

(User) Properties

Subscription IDs

Reason Codes

Enhanced Authentication

Server reference

Request / Response

Server disconnect

Reason Strings

Assigned Client ID

Server Keep Alive

Server feature / capability management

MQTTv5: Properties everywhere

› Configurability, Flexibility, Control
› Minimal cost (in message size)

CONNECT	SUBSCRIBE	PUBLISH
Session Expiry	Subscription ID	Payload Format
Receive Maximum	User Properties	Message Expiry
Maximum Packet Size		Topic Alias
Topic Alias Maximum		Response Topic
Request Response Information		Correlation Data
Request Problem Information		User Properties
User Properties		Subscription ID
Auth. Method		Content Type
Auth. Data		

Packet Layout (MQTTv5)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Connect	Length		“MQTT”				Version	Flags	Keep Alive		PropSize	Properties		Payload
Subscribe	Length		ID		Topic Length		“a/b”			QoS	PropSize	Properties		
Publish	Length		Topic Length		“a/b”			Packet ID		PropSize	Properties		Payload	

- Properties are added at the end of the variable header
- Property Size is variable byte integer (size 1-4)
- Properties are designed as a set (type id, value)
  - Example: 0x27 (Max Packet Size) 65535 (as 4 byte integer)
- Minimum additional overhead is 1 byte
  - Indicating no properties set

MQTTv5: Features for embedded

- Connectivity Limitations
  - Session Expiry
  - Message Expiry
  - Will Delay
- Hardware Limitations
  - Maximum Packet Size
- Bandwidth Limitations
  - Topic Alias

MQTTv5: Topic Alias

- Example: Sensor Network
  - Topic: sensors/Europe/2af89b42-d2a6-11e8-a8d5-f2801f1b9fd1/Temperature
- Topic **must** be part of **every** message publication
  - Topic 63 bytes plus size description 1 byte => 64 bytes

PUBLISH (1)
Msg Length (1): 69
Topic Size (1): 63
Topic (63): sensors/Europe/2af89b…
Properties (1): 0
Payload (4): "28.4"

71 bytes per msg

PUBLISH (1)
Msg Length (1): 72
Topic Size (1): 63
Topic (63): sensors/Europe/2af89b…
Properties (1): 2
Prop Type(1): 0x23
Prop Value(2): 1
Payload (4): "28.4"

74 bytes (first)

PUBLISH (1)
Msg Length (1): 9
Topic Size (1): 0
Properties (1): 2
Prop Type(1): 0x23
Prop Value(2): 1
Payload (4): "26.2"

11 bytes (following)

MQTTv5: Authentication / Authorization

- MQTT 3.1.1 relied on
  - TLS on transport level
  - Username / password authentication on connect
- Caused server providers to create custom solutions
- Users wanted more flexibility
  - Specify authentication methods (preferably pluggable)
  - More fine-grained authorization (e.g. Topic access)

Available commands

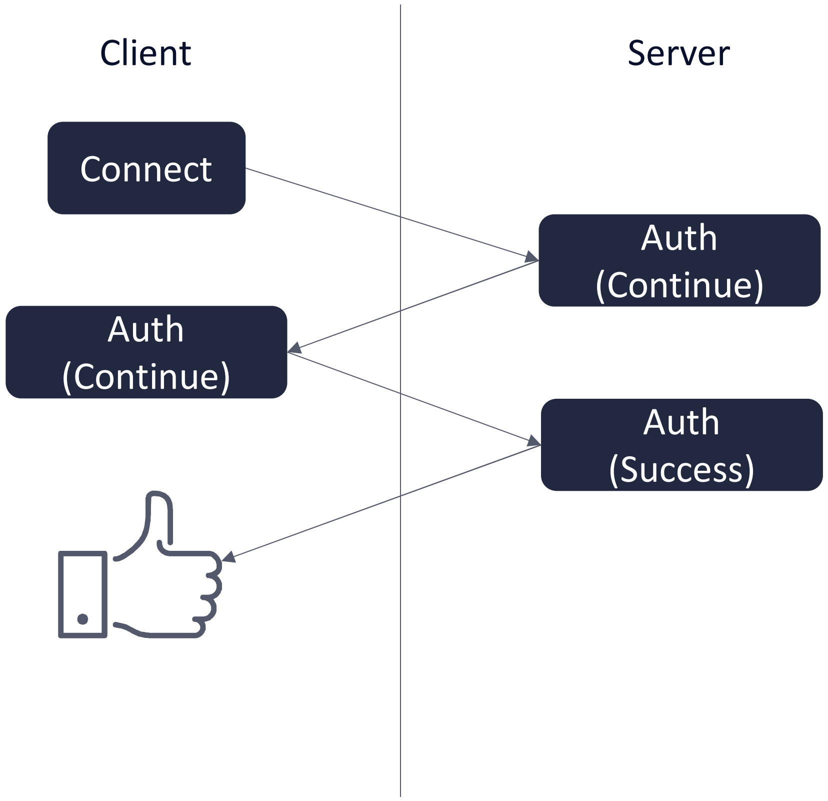
Command	Value	Description	Direction
CONNECT	0x10	Request Connection	C->S
CONNACK	0x20	Connection request accepted	S->C
PUBLISH	0x30	Send/Receive message	C->S
PUBACK	0x40	Message has been received and handled by server (QoS1)	S->C
PUBREC	0x50	Message has been received (QoS2)	S->C
PUBREL	0x60	Message can be released (QoS2)	C->S
PUBCOMP	0x70	Message handling completed (QoS2)	S->C
SUBSCRIBE	0x80	Subscribe to one or more topics	C->S
SUBACK	0x90	Subscription request has been accepted	S->C
UNSUBSCRIBE	0xA0	Remove subscription	C->S
UNSUBACK	0xB0	Subscription removal done	S->C
PINGREQ	0xC0	Ping	C->S
PINGRESP	0xD0	Pong	S->C
DISCONNECT	0xE0	Request Clean Disconnect	C->S

Available commands (MQTTv5)

Command	Value	Description	Direction
CONNECT	0x10	Request Connection	C->S
CONNACK	0x20	Connection request accepted	S->C
PUBLISH	0x30	Send/Receive message	C->S
PUBACK	0x40	Message has been received and handled by server (QoS1)	S->C
PUBREC	0x50	Message has been received (QoS2)	S->C
PUBREL	0x60	Message can be released (QoS2)	C->S
PUBCOMP	0x70	Message handling completed (QoS2)	S->C
SUBSCRIBE	0x80	Subscribe to one or more topics	C->S
SUBACK	0x90	Subscription request has been accepted	S->C
UNSUBSCRIBE	0xA0	Remove subscription	C->S
UNSUBACK	0xB0	Subscription removal done	S->C
PINGREQ	0xC0	Ping	C->S
PINGRESP	0xD0	Pong	S->C
DISCONNECT	0xE0	Request Clean Disconnect	C->S
AUTH	0xF0	Authentication	C<->S

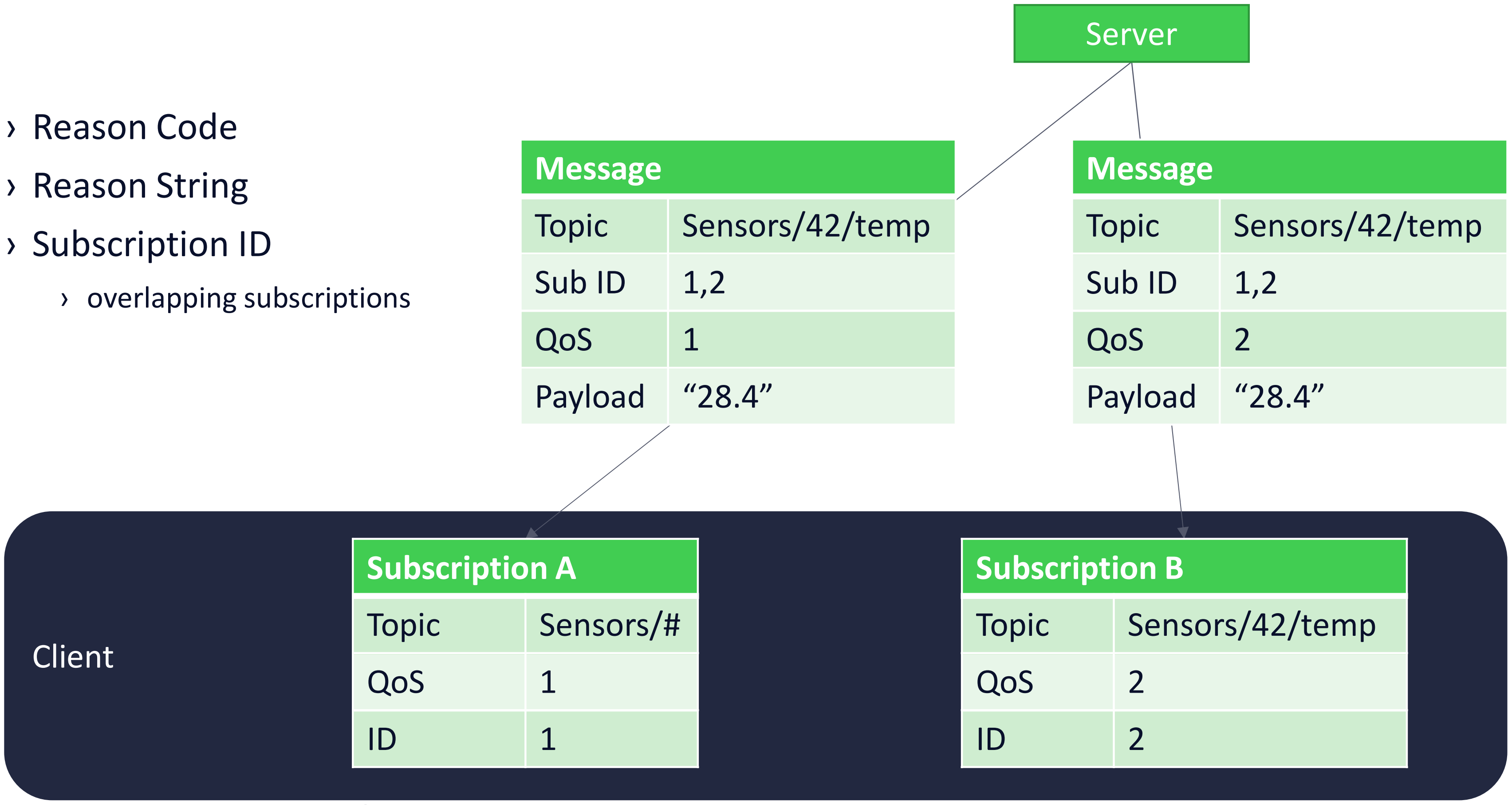
MQTTv5: Authentication

- Bidirectional Command
  - Contains reason code to indicate auth state
    - 0x00: Auth Success
    - 0x18: Continue Authentication
    - 0x19: Re-authenticate
- Properties
  - Authentication Method
  - Authentication Data
  - Reason String
  - User Properties
- Payload



MQTTv5: Error handling / debugging

- Reason Code
- Reason String
- Subscription ID
  - overlapping subscriptions



Message	
Topic	Sensors/42/temp
Sub ID	1,2
QoS	1
Payload	“28.4”

Message	
Topic	Sensors/42/temp
Sub ID	1,2
QoS	2
Payload	“28.4”

Subscription A	
Topic	Sensors/#
QoS	1
ID	1

Subscription B	
Topic	Sensors/42/temp
QoS	2
ID	2

Downsides of MQTT (PubSub in general)

- High transport requirements
  - Ordered, lossless, bi-directional
  - TCP mostly used, if not only, approach
- Server is the bottleneck
  - Clusters, Bridges
  - Load-balancing
- No RPC
- No 1:1 connection

Considered for MQTTv6, (potentially use) MQTT-SN

Not designed to do so

Available solutions (MQTTv5)

- Client
  - Qt MQTT
    - C++, https://github.com/qt/qtmqtt
  - gmqtt
    - Python, https://github.com/wialon/gmqtt
  - Zotonic mqtt_packet_map
    - Erlang, https://github.com/zotonic/mqtt_packet_map
- Server
  - Flespi
  - Vernemq
  - Eclipse Paho Testing Utilities

Food for thought: Servers and Embedded

- MQTT Servers are in the cloud or on the edge
- For high level embedded (ARM64, …) containers are getting more traction
  - Security
  - OTA
  - …
- Data synchronization / Telemetry between containers
  - MQTT

Thank you

Resources:

- OASIS TC https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt
- https://www.mqtt.org
- https://www.youtube.com/watch?v=CJX-x24NVqs Ian Craggs MQTT5

Maurice Kalinowski, Maurice.Kalinowski@qt.io @maukalinow