

# ***Mysteries of IOT Revealed***

**Gary Dischner TxMqQ**

# Agenda

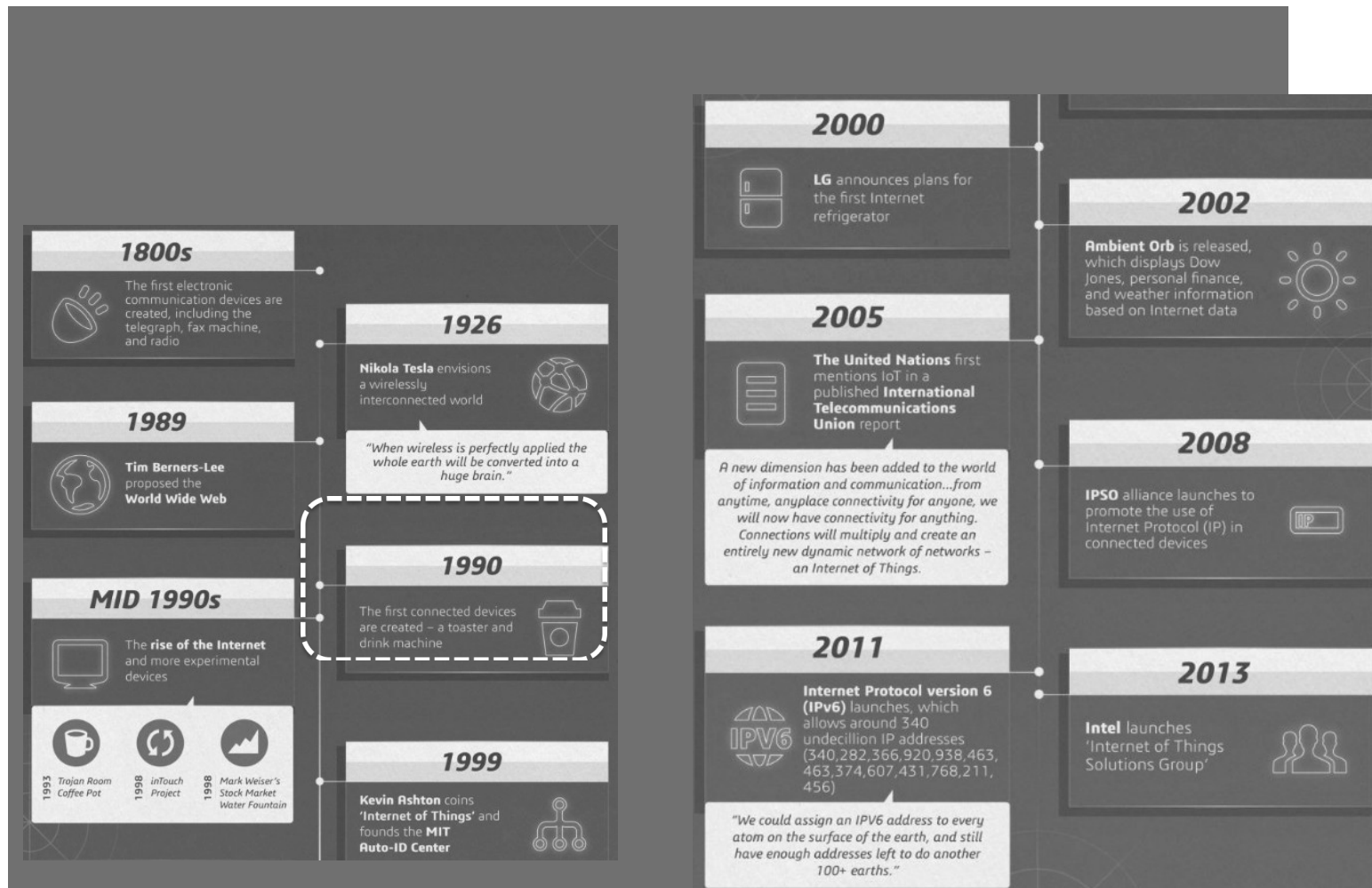
- What is IOT?
- Characteristics of IOT
- What is MQTT?
- Characteristics of MQTT

# Agenda

## ■ What is IOT?



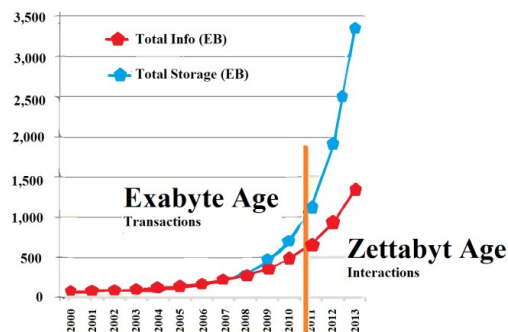
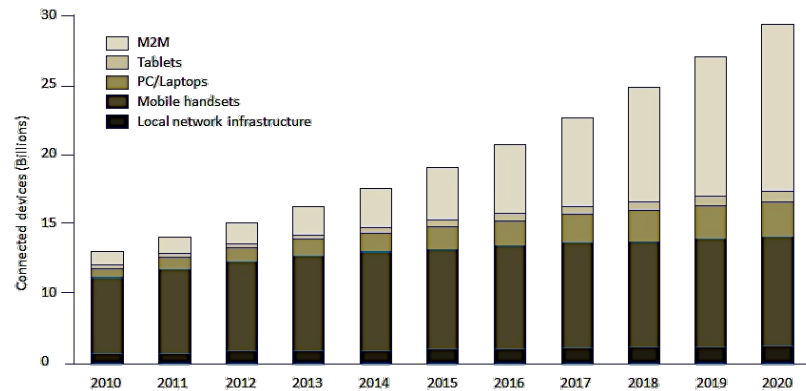
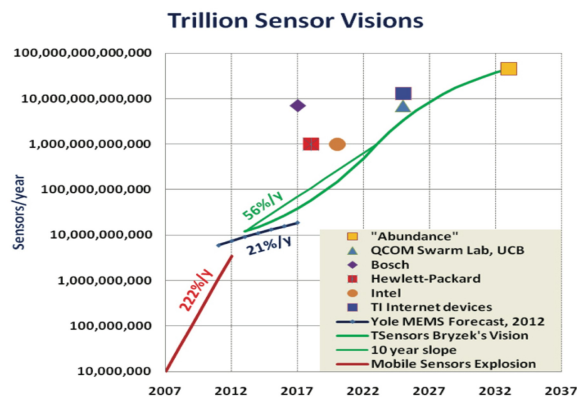
# History of Internet of Things





# Internet of Things Wave

Expected growth in the number of connected devices

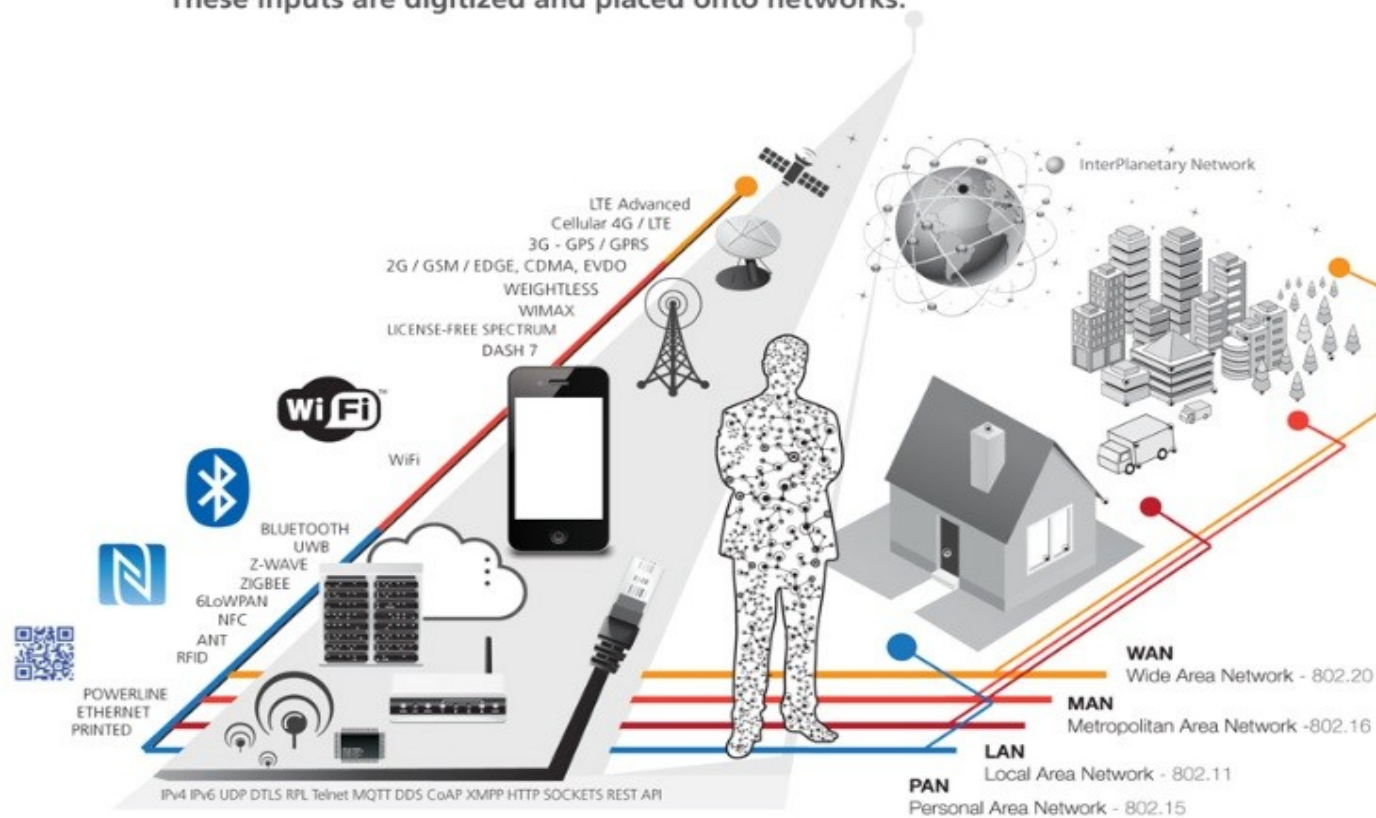


## Definition of IoT\* :

Intelligent interactivity between human and things to exchange information & knowledge for new value creation.

## 2 CONNECTIVITY

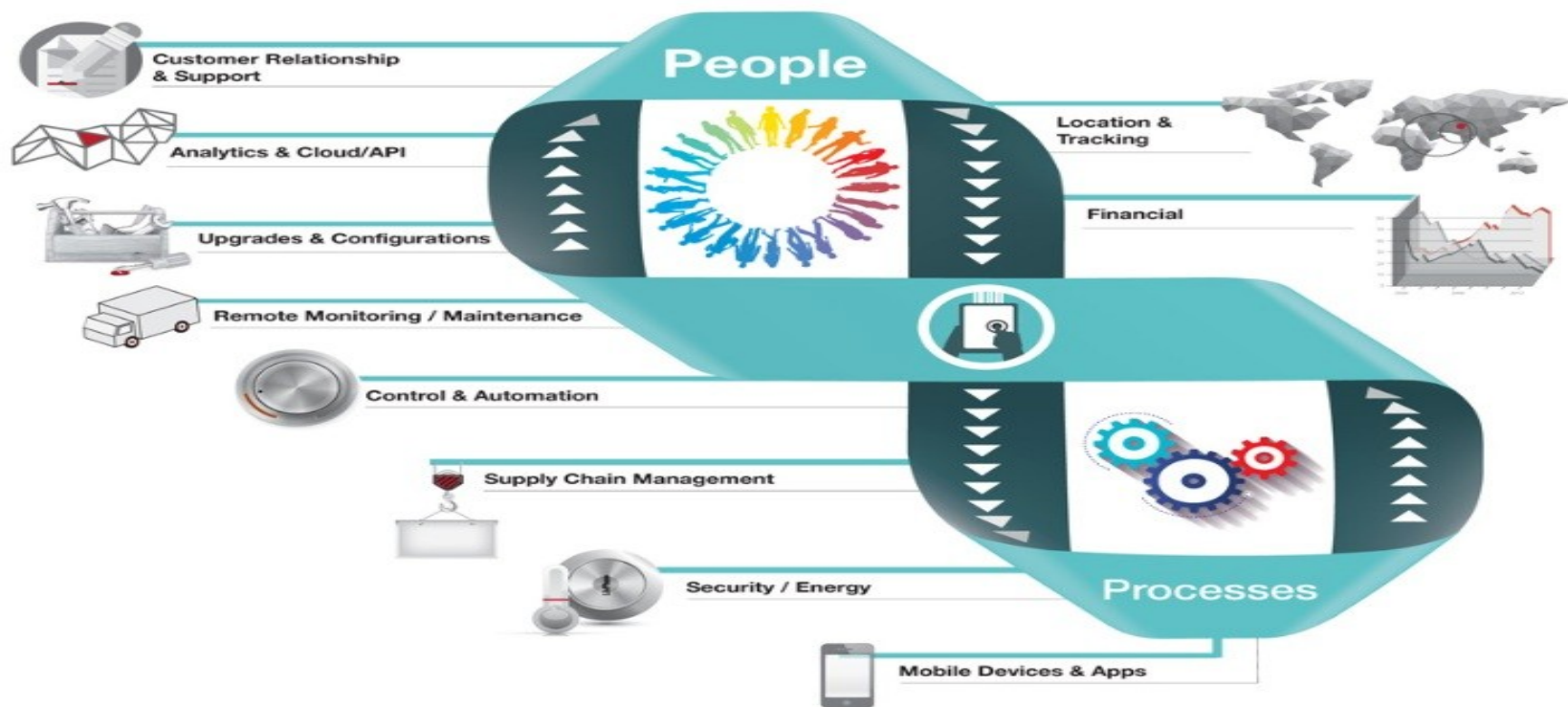
These inputs are digitized and placed onto networks.



[Source: Postscape - <http://postscapes.com/what-exactly-is-the-internet-of-things-infographic>.]

### 3 PEOPLE & PROCESSES

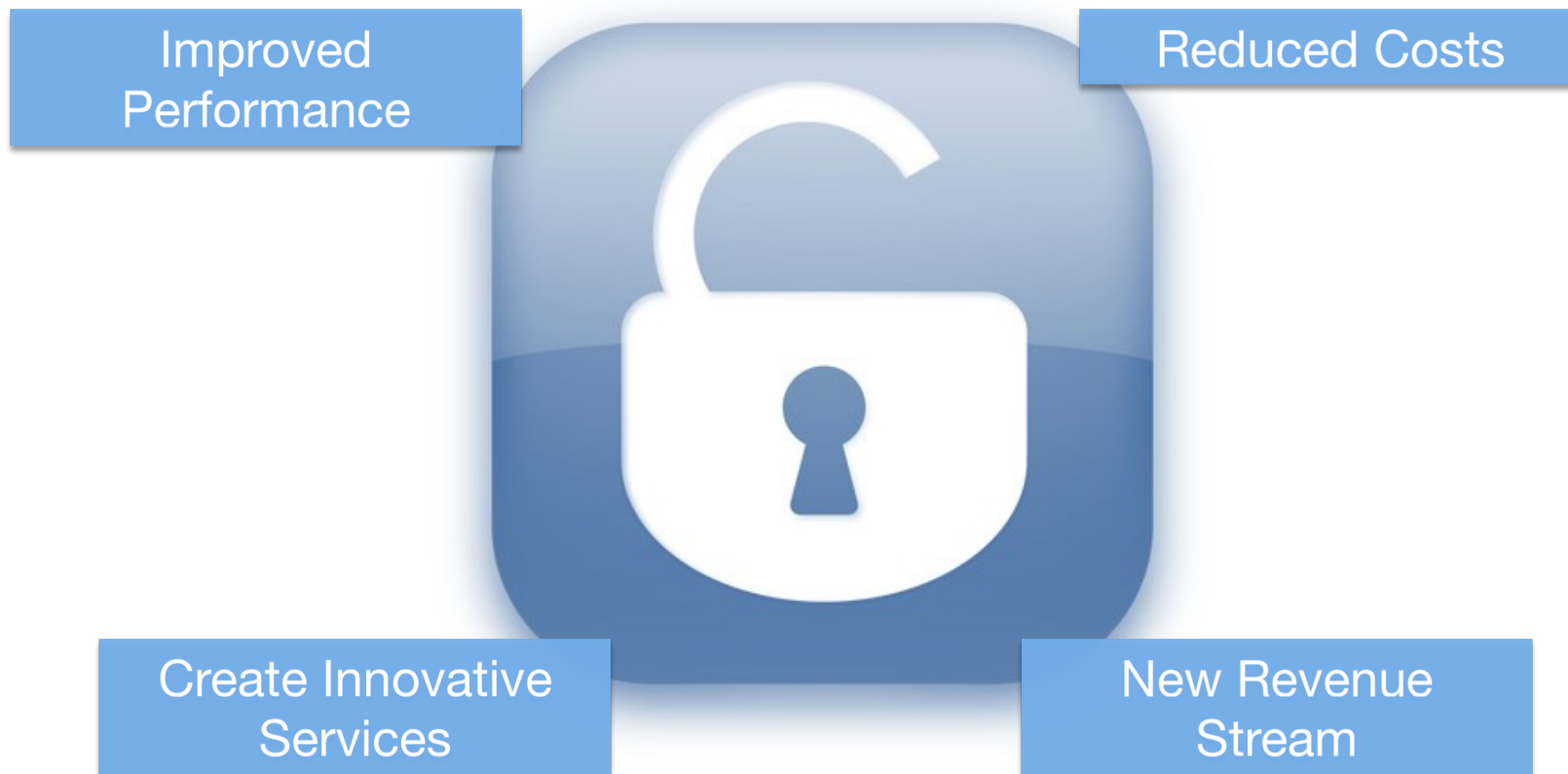
These networked inputs can then be combined into bi-directional systems that integrate data, people, processes and systems for better decision making.





# ■ Characteristics of IOT

# Unlocking the Massive Potential of IoT





Convergence  
of Technology  
Trends



IOT Applications

# The interactions between these entities are creating new types of smart applications and services.

SENSORS + CONNECTIVITY + PEOPLE + PROCESSES

Starting with popular connected devices already on the market



## SMART THERMOSTATS



Save resources and money on your heating bills by adapting to your usage patterns and turning the temperature down when you're away from home.

## CONNECTED CARS



Tracked and rented using a smartphone. Car2Go also handles billing, parking and insurance automatically.

## ACTIVITY TRACKERS



Continuously capture heart rate patterns, activity levels, calorie expenditure and skin temperature on your wrist 24/7.

## SMART OUTLETS



Remotely turn any device or appliance on or off. Track a device's energy usage and receive personalized notifications from your smartphone.

## PARKING SENSORS



Using embedded street sensors, users can identify real-time availability of parking spaces on their phone. City officials can manage and price their resources based on actual use.

[Source: Postscapes - <http://postscapes.com/what-exactly-is-the-internet-of-things-infographic>.]

# IoT protocols war!

AMQP

HTTP

CoA  
P

MQTT

XMPP

DDS

STOMP





MQTT: a protocol for collecting device data and communicating it to servers (D2S)

- XMPP: a protocol best for connecting devices to people, a special case of the D2S pattern, since people are connected to the servers
- DDS: a fast bus for integrating intelligent machines (D2D)
- AMQP: a queuing system designed to connect servers to each other (S2S)

# ■ What is MQTT

# MQTT – MQ Telemetry Transport

## 1. What is MQTT?

MQTT is a lightweight message queueing and transport protocol.

MQTT, as its name implies, is suited for the transport of telemetry data (sensor and actor data).

MQTT is very lightweight and thus suited for M2M (Mobile to Mobile), WSN (Wireless Sensor Networks) and ultimately IoT (Internet of Things) scenarios where sensor and actor nodes communicate with applications through the MQTT message broker.

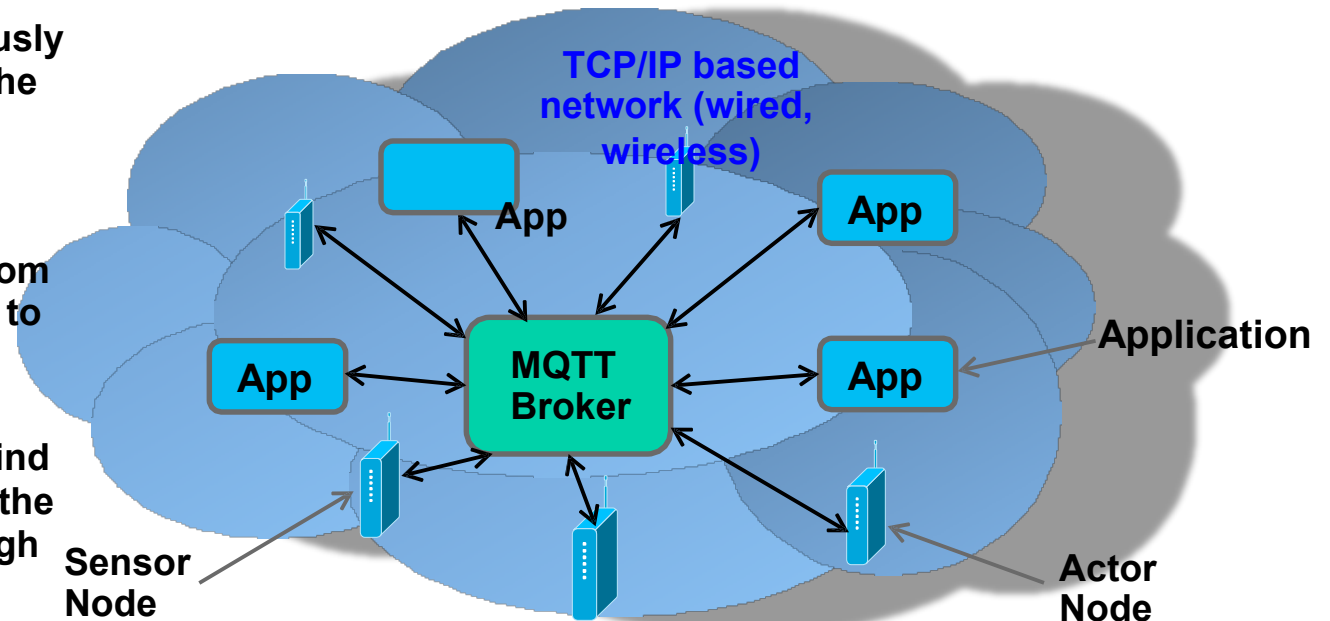
### Example:

**Light sensor** continuously sends sensor data to the broker.

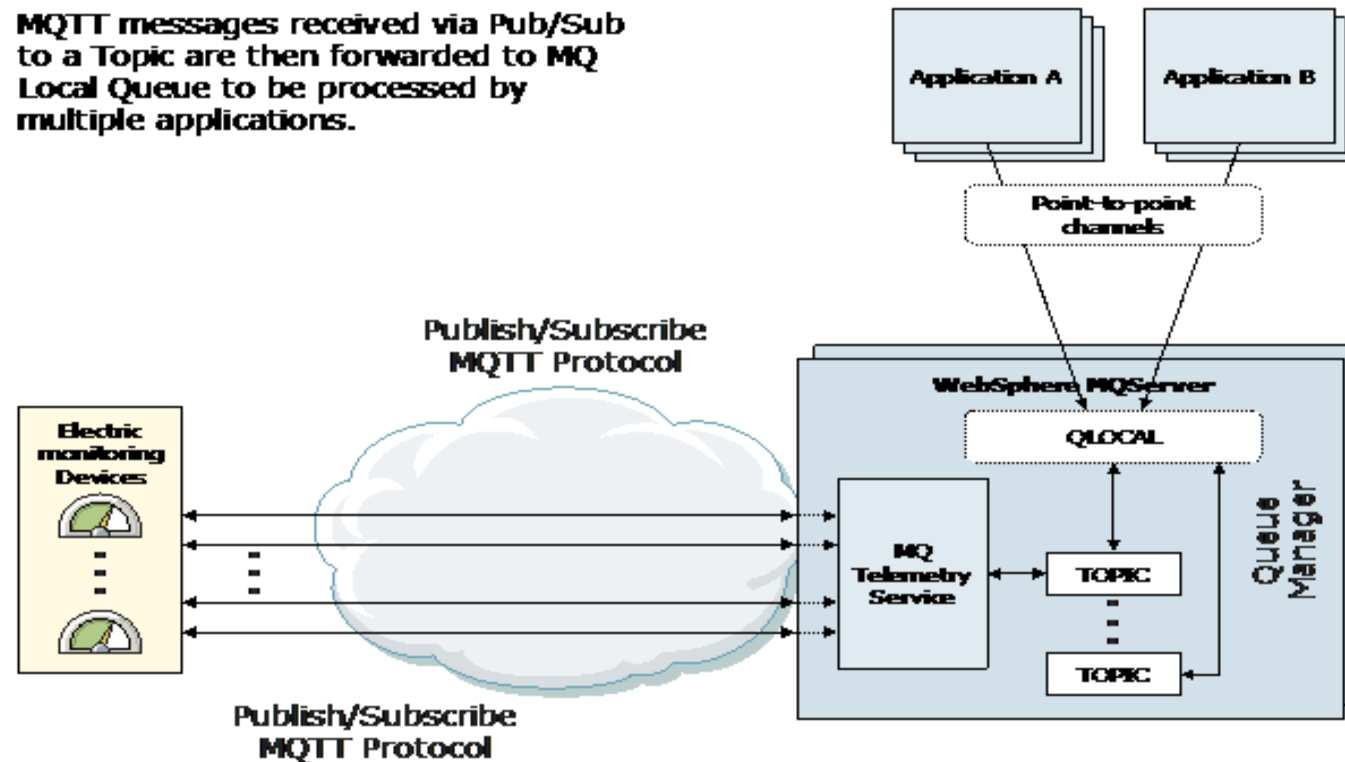
### **Building control application**

receives sensor data from the broker and decides to activate the blinds.

Application sends a blind activation message to the **blind actor node** through the broker.



MQTT messages received via Pub/Sub to a Topic are then forwarded to MQ Local Queue to be processed by multiple applications.



# ■MQTT Characteristics

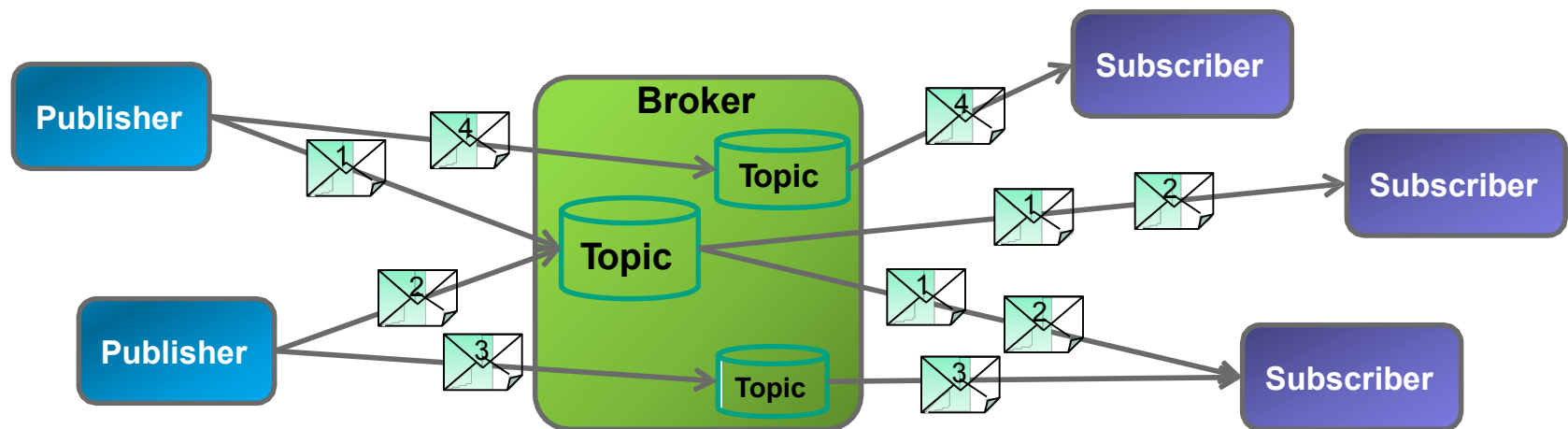


# MQTT – MQ Telemetry Transport

## 2. MQTT characteristics

### MQTT Key features:

- Lightweight message queueing and transport protocol
- Asynchronous communication model with messages (events)
- Low overhead (2 bytes header) for low network bandwidth applications
- Publish / Subscribe (PubSub) model
- Decoupling of data producer (publisher) and data consumer (subscriber) through topics (message queues)
- Simple protocol, aimed at low complexity, low power and low footprint implementations (e.g. WSN - Wireless Sensor Networks)
- Runs on connection-oriented transport (TCP). To be used in conjunction with 6LoWPAN (TCP header compression)
- MQTT caters for (wireless) network disruptions



# MQTT – MQ Telemetry Transport

## 3. Origins and future of MQTT standard

### The past, present and future of MQTT:

MQTT was developed by IBM and Eurotech.

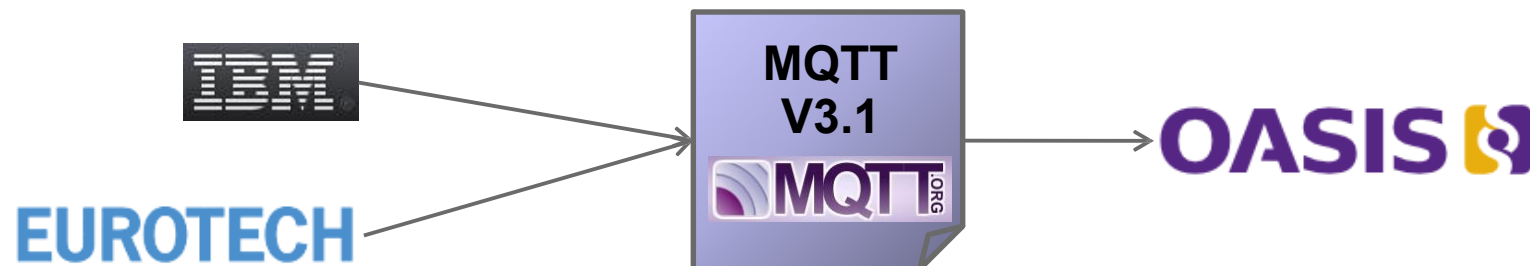
The current version 3.1 is available from <http://mqtt.org/>.

Eventually, MQTT version 3.1 is adopted and published as an official standard by OASIS

As such, OASIS became the new home for the development of MQTT.

The OASIS TC (Technical Committee) tasked with the further development of MQTT commits to the following:

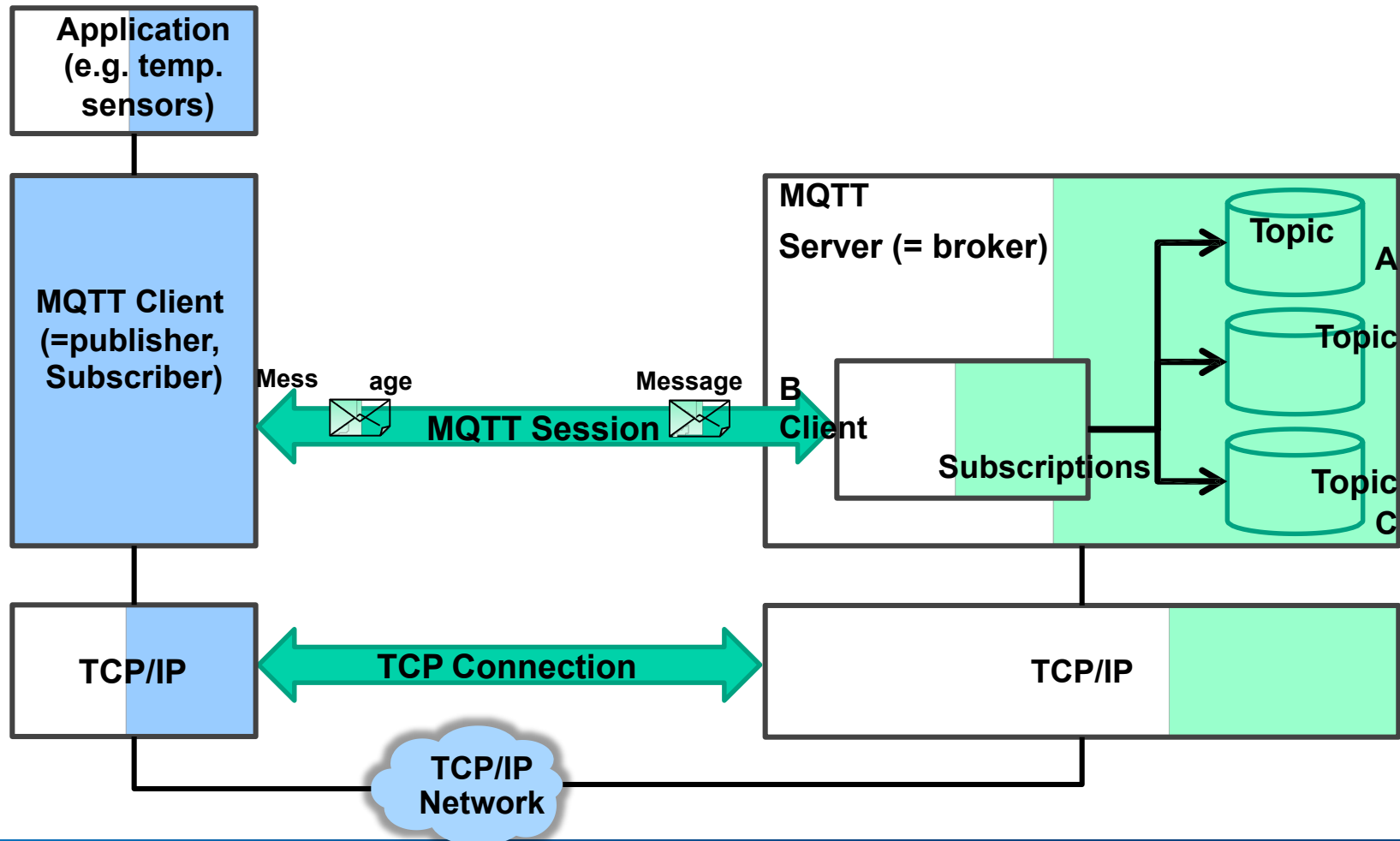
- Backward compatibility of forthcoming OASIS MQTT standard with MQTT V3.1
- Changes restricted to the CONNECT message
- Clarification of existing version V3.1



# MQTT – MQ Telemetry Transport

## 4. MQTT model (1/3)

The core elements of MQTT are clients, servers (=brokers), sessions, subscriptions and topics.

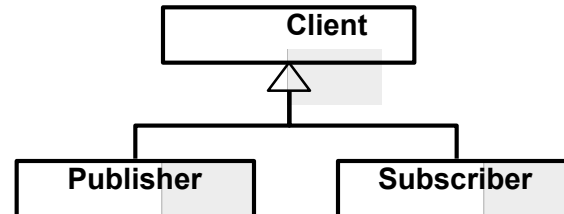


# MQTT – MQ Telemetry Transport

## 4. MQTT model (2/3)

MQTT client (=publisher, subscriber):

Clients subscribe to topics to publish and receive messages.  
Thus subscriber and publisher are special roles of a client.



MQTT server (=broker):

Servers run topics, i.e. receive subscriptions from clients on topics, receive messages from clients and forward these, based on client's subscriptions, to interested clients.

Topic:

Technically, topics are message queues. Topics support the publish/subscribe pattern for clients.

Logically, topics allow clients to exchange information with defined semantics. Example topic: Temperature sensor data of a building.



# MQTT – MQ Telemetry Transport

## 4. MQTT model (3/3)

### Session:

A session identifies a (possibly temporary) attachment of a client to a server. All communication between client and server takes place as part of a session.

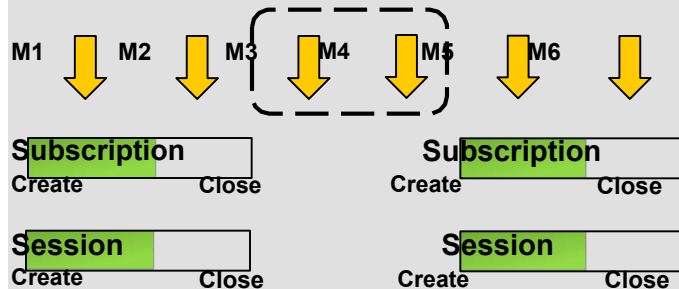
### Subscription:

Unlike sessions, a subscription logically attaches a client to a topic. When subscribed to a topic, a client can exchange messages with a topic.

Subscriptions can be «transient» or «durable», depending on the clean session flag in the CONNECT message:

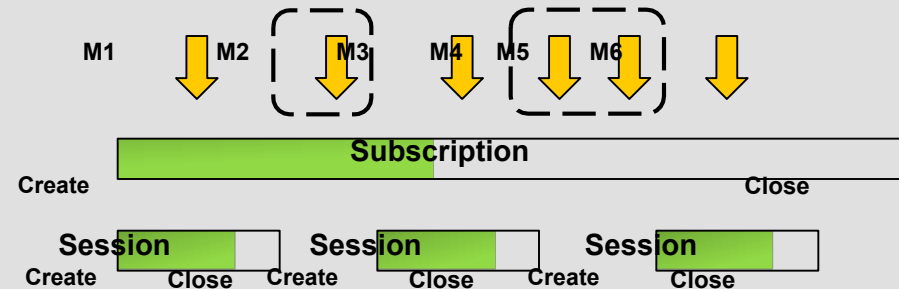
#### «Transient» subscription ends with session:

Messages M3 and M4 are not received by the client as soon as it creates / opens a new session.



#### «Durable» subscription:

Messages M2, M4 and M5 are not lost but will be received by the



### Message:

Messages are the units of data exchange between topic clients.

MQTT is agnostic to the internal structure of messages.



# MQTT – MQ Telemetry Transport

## 5. MQTT message format (1/14)

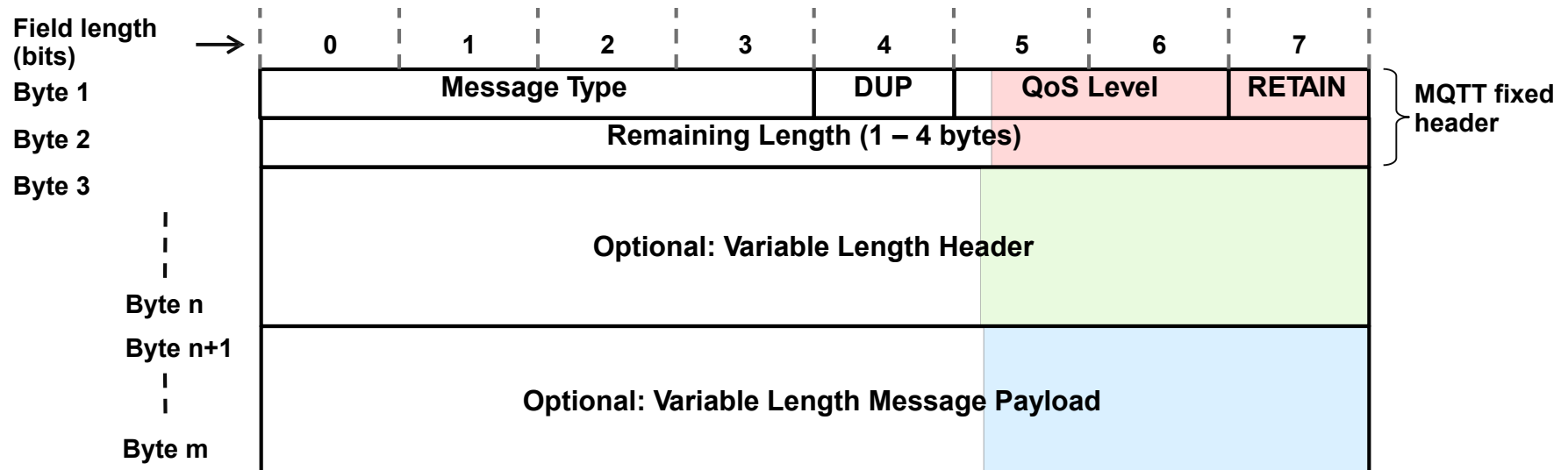
### Message format:

MQTT messages contain a mandatory fixed-length header (2 bytes) and an optional message-specific variable length header and message payload.

Optional fields usually complicate protocol processing.

However, MQTT is optimized for bandwidth constrained and unreliable networks (typically wireless networks), so optional fields are used to reduce data transmissions as much as possible.

MQTT uses network byte and bit ordering.



# MQTT – MQ Telemetry Transport

## 5. MQTT message format (2/14)

### Overview of fixed header fields:

Message fixed header field	Description / Values	
Message Type	0: Reserved	8: SUBSCRIBE
	1: CONNECT	9: SUBACK
	2: CONNACK	10: UNSUBSCRIBE
	3: PUBLISH	11: UNSUBACK
	4: PUBACK	12: PINGREQ
	5: PUBREC	13: PINGRESP
	6: PUBREL	14: DISCONNECT
	7: PUBCOMP	15: Reserved
DUP	Duplicate message flag. Indicates to the receiver that this message may have already been received. 1: Client or server (broker) re-delivers a PUBLISH, PUBREL, SUBSCRIBE or UNSUBSCRIBE message (duplicate message).	
QoS Level	Indicates the level of delivery assurance of a PUBLISH message. 0: At-most-once delivery, no guarantees, «Fire and Forget». 1: At-least-once delivery, acknowledged delivery. 2: Exactly-once delivery. Further details see <a href="#">MQTT QoS</a> .	
RETAIN	1: Instructs the server to retain the last received PUBLISH message and deliver it as a first message to new subscriptions. Further details see <a href="#">RETAIN (keep last message)</a> .	
Remaining Length	Indicates the number of remaining bytes in the message, i.e. the length of the (optional) variable length header and (optional) payload. Further details see <a href="#">Remaining length (RL)</a> .	

# MQTT – MQ Telemetry Transport

## 5. MQTT message format (3/14)

### RETAIN (keep last message):

RETAIN=1 in a PUBLISH message instructs the server to keep the message for this topic. When a new client subscribes to the topic, the server sends the retained message.

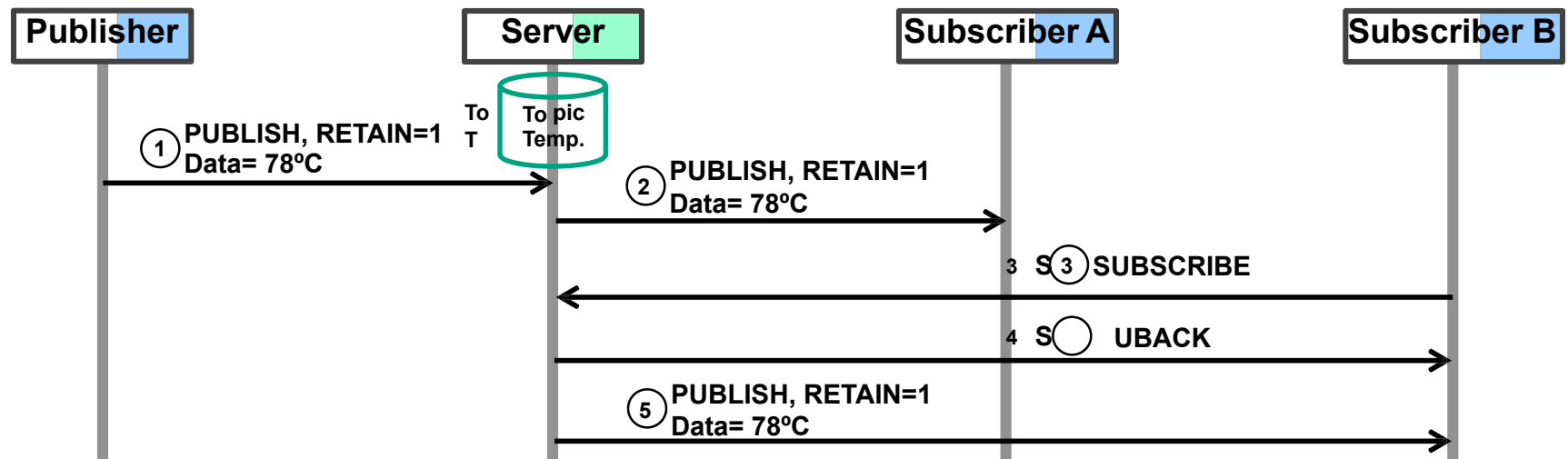
Typical application scenarios:

Clients publish only changes in data, so subscribers receive the **last known good value**.

Example:

Subscribers receive last known temperature value from the temperature data topic.

RETAIN=1 indicates to subscriber B that the message may have been published some time ago.



# MQTT – MQ Telemetry Transport

## 5. MQTT message format (4/14)

### Remaining length (RL):

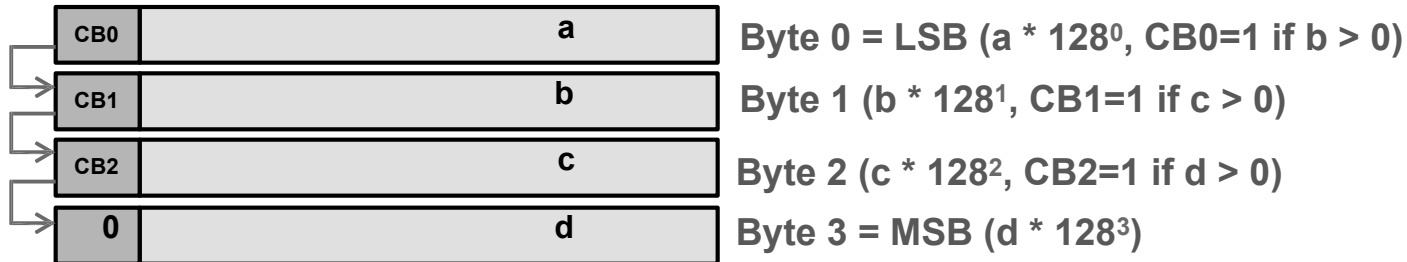
The remaining length field encodes the sum of the lengths of:

- a. (Optional) variable length header
- b. (Optional) payload

To save bits, remaining length is a variable length field with 1...4 bytes.

The most significant bit of a length field byte has the meaning «continuation bit» (CB). If more bytes follow, it is set to 1.

Remaining length is encoded as  $a * 128^0 + b * 128^1 + c * 128^2 + d * 128^3$  and placed into the RL field bytes as follows:



**Key:**  
LSB: Least Significant Byte  
MSB: Most Significant Byte

**Example 1:**  $RL = 364 = 108 * 128^0 + 2 * 128^1 \rightarrow a=108, CB0=1, b=2, CB1=0, c=0, d=0, CB2=0$

**Example 2:**  $RL = 25'897 = 41 * 128^0 + 74 * 128^1 + 1 * 128^2 \rightarrow a=41, CB0=1, b=74, CB1=1, c=1, CB2=0, d=0$

# MQTT for Sensor Networks






# MQTT vs MQTT-S

	MQTT	MQTT-S
Transport type	Reliable point to point datagrams	Unreliable point streams
Communication	TCP/IP	Non-IP or UDP
Networking Bluetooth, RF	Ethernet, WiFi, 3G	ZigBee,
Min message size	2 bytes - PING	1 byte
Max message size (*)	≤ 24MB	< 128 bytes
Battery-operated		√
Sleeping clients		√
QoS: -1 “dumb client”		√

QoS level	Message delivery	Delivery semantics	Delivery Guarantees
-1* setup	$\leq 1$	At most once	No connection  Transmit only
Best effort – no guarantees (*) - MQTT-S only			
0 No guarantees	$\leq 1$	At most once	Best effort
1 delivery Duplicates possible	$\geq 1$	At least once	Guaranteed
2 delivery	$\equiv 1$	Exactly once	Guaranteed  No duplicates



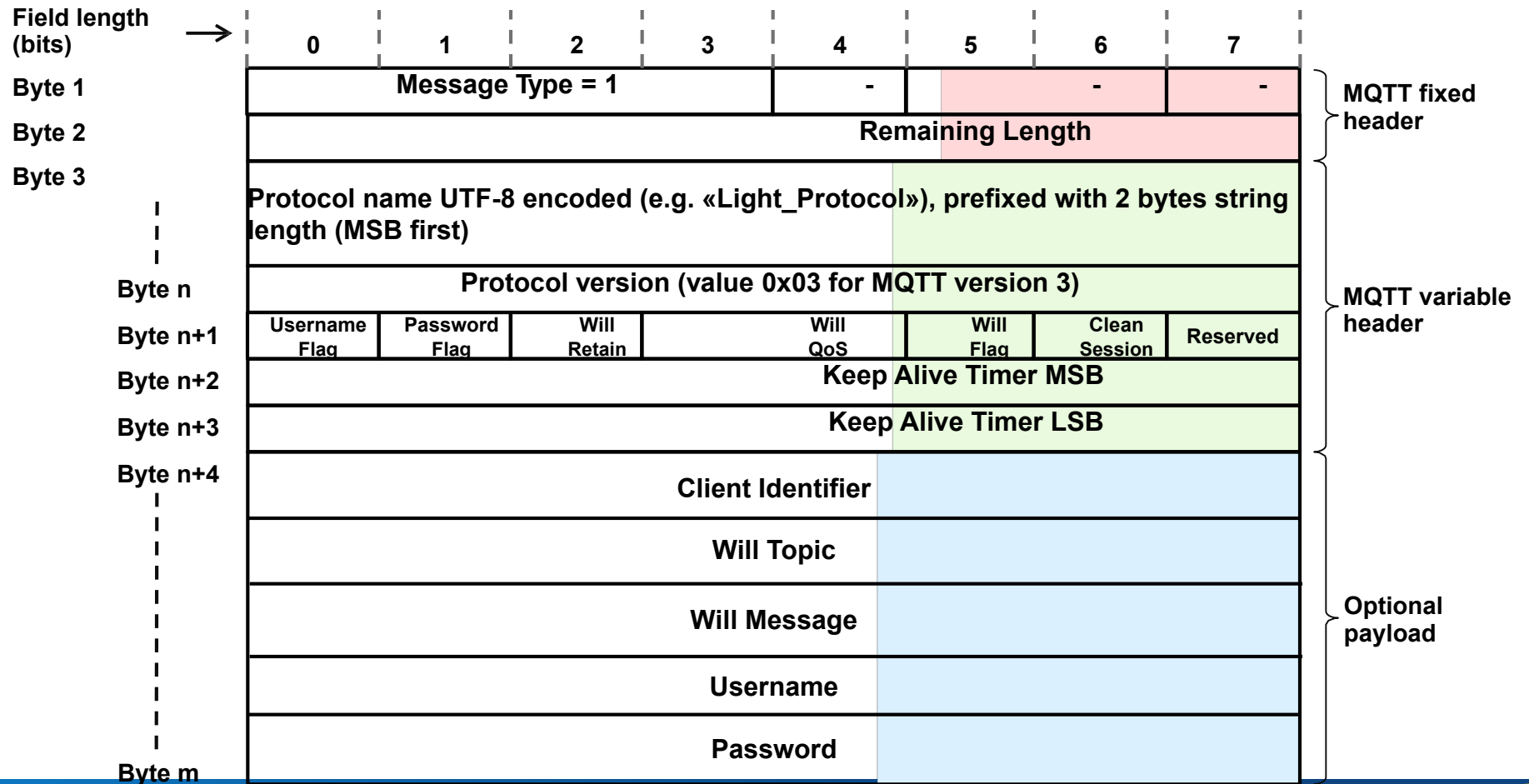
**CONNECT 1 Client request to connect to Server**  
**CONNACK 2 Connect Acknowledgment**  
**PUBLISH 3 Publish message**  
**PUBACK 4 Publish Acknowledgment**  
**PUBREC 5 Publish Received (assured delivery part 1)**  
**PUBREL 6 Publish Release (assured delivery part 2)**  
**PUBCOMP 7 Publish Complete (assured delivery part 3)**  
**SUBSCRIBE 8 Client Subscribe request**  
**SUBACK 9 Subscribe Acknowledgment**  
**UNSUBSCRIBE 10 Client Unsubscribe request**  
**UNSUBACK 11 Unsubscribe Acknowledgment**  
**PINGREC 12 PING Request**  
**PINGRESP 13 PING Response**  
**DISCONNECT 14 Client is Disconnecting**

# MQTT – MQ Telemetry Transport

## 5. MQTT message format (5/14)

### CONNECT message format:

The CONNECT message contains many session-related information as optional header fields.



# MQTT – MQ Telemetry Transport

## 5. MQTT message format (6/14)

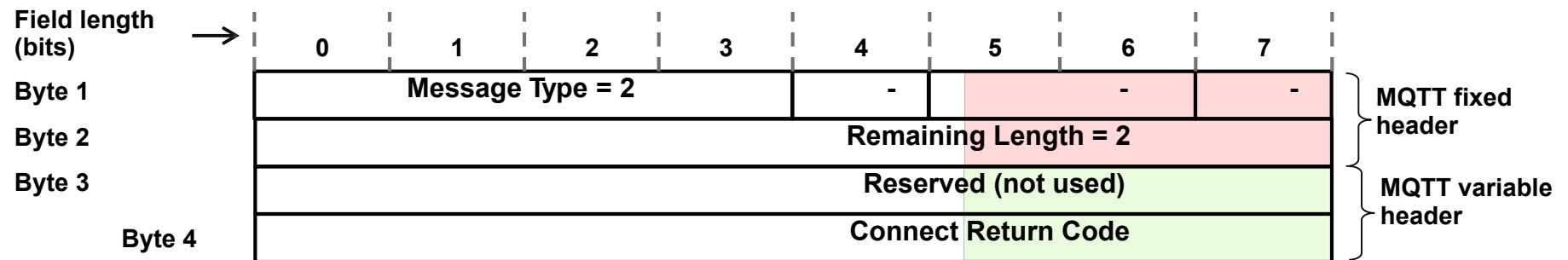
### Overview CONNECT message fields:

CONNECT message field	Description / Values
Protocol Name	UTF-8 encoded protocol name string. Example: «Light_Protocol»
Protocol Version	Value 3 for MQTT V3.
Username Flag	If set to 1 indicates that payload contains a username.
Password Flag	If set to 1 indicates that payload contains a password. If username flag is set, password flag and password must be set as well.
Will Retain	If set to 1 indicates to server that it should retain a Will message for the client which is published in case the client disconnects unexpectedly.
Will QoS	Specifies the QoS level for a Will message.
Will Flag	Indicates that the message contains a Will message in the payload along with Will retain and Will QoS flags. More details see <a href="#">MQTT will message</a> .
Clean Session	If set to 1, the server discards any previous information about the (re)-connecting client (clean new session). If set to 0, the server keeps the subscriptions of a disconnecting client including storing QoS level 1 and 2 messages for this client. When the client reconnects, the server publishes the stored messages to the client.
Keep Alive Timer	Used by the server to detect broken connections to the client. More details see <a href="#">Keepalive timer</a> .
Client Identifier	The client identifier (between 1 and 23 characters) uniquely identifies the client to the server. The client identifier must be unique across all clients connecting to a server.
Will Topic	Will topic to which a will message is published if the will flag is set.
Will Message	Will message to be published if will flag is set.
Username and Password	Username and password if the corresponding flags are set.

# MQTT – MQ Telemetry Transport

## 5. MQTT message format (7/14)

### CONNACK message format:

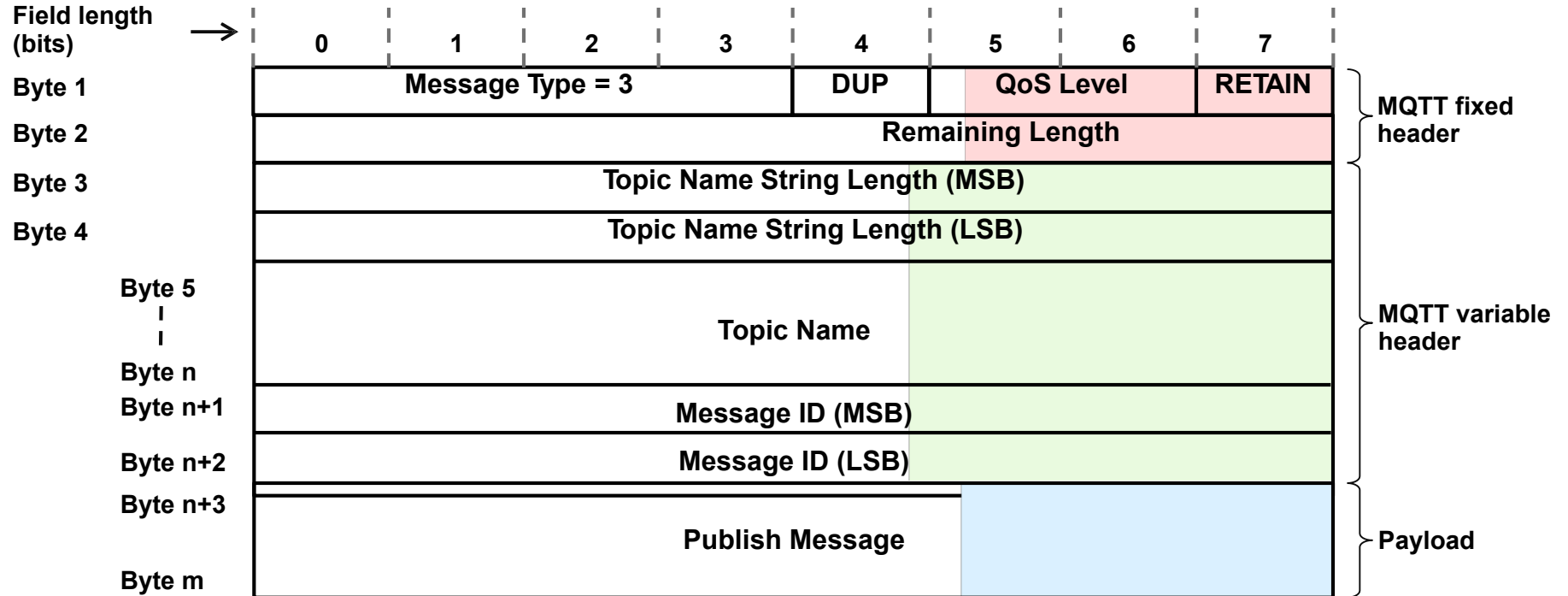


CONNACK message field	Description / Values
Reserved	Reserved field for future use.
Connect Return Code	0: Connection Accepted 1: Connection Refused, reason = unacceptable protocol version 2: Connection Refused, reason = identifier rejected 3: Connection Refused, reason = server unavailable 4: Connection Refused, reason = bad user name or password 5: Connection Refused, reason = not authorized 6-255: Reserved for future use

# MQTT – MQ Telemetry Transport

## 5. MQTT message format (8/14)

### PUBLISH message format:

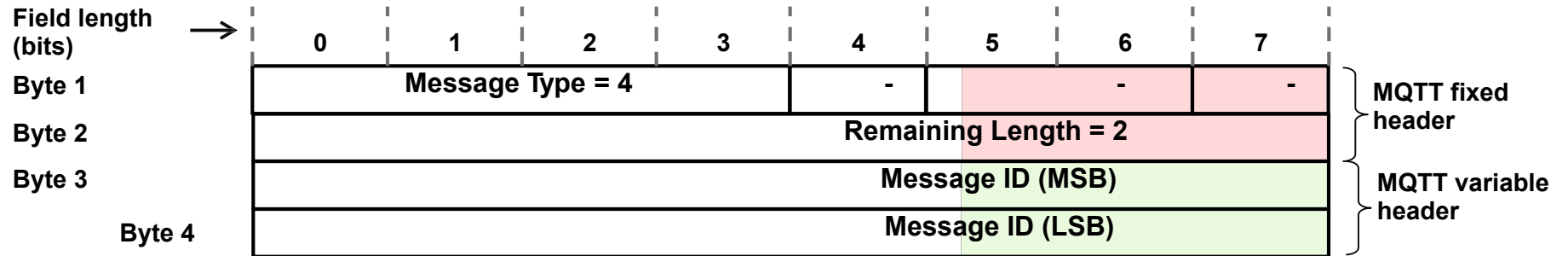


PUBLISH message field	Description / Values
Topic Name with Topic Name String Length	Name of topic to which the message is published. The first 2 bytes of the topic name field indicate the topic name string length.
Message ID	A message ID is present if QoS is 1 (At-least-once delivery, acknowledged delivery) or 2 (Exactly-once delivery).
Publish Message	Message as an array of bytes. The structure of the publish message is application-specific.

# MQTT – MQ Telemetry Transport

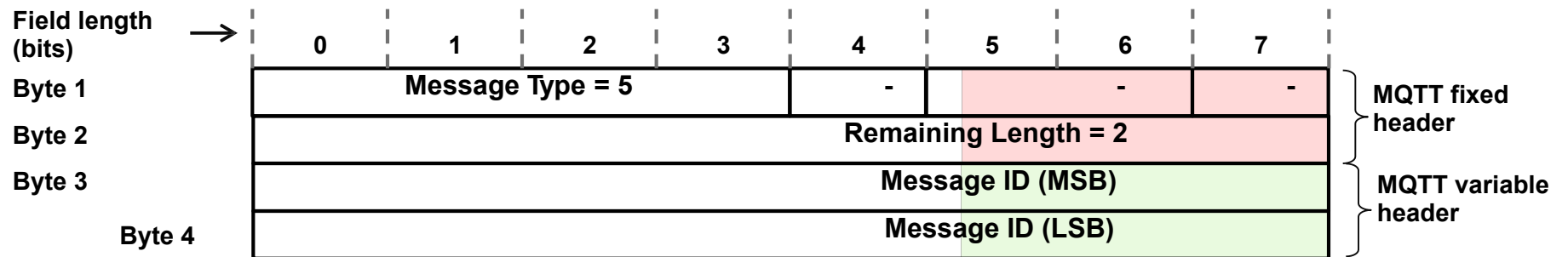
## 5. MQTT message format (9/14)

### PUBACK message format:



PUBACK message field	Description / Values
Message ID	The message ID of the PUBLISH message to be acknowledged.

### PUBREC message format:



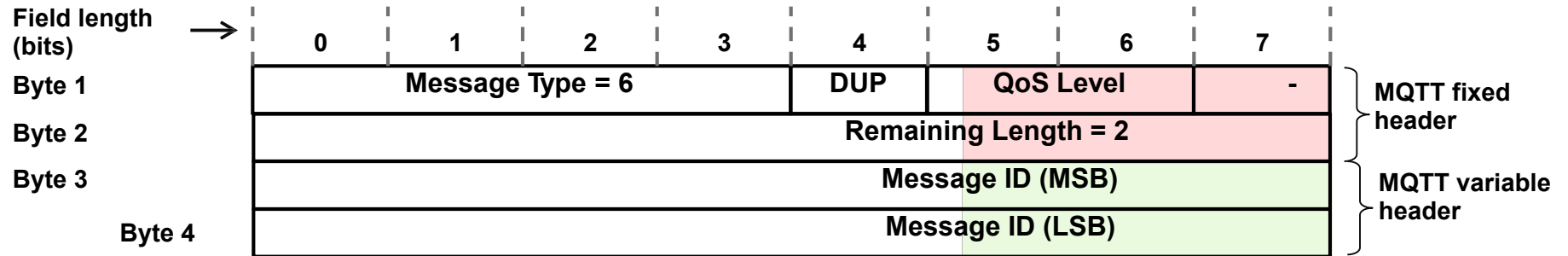
PUBREC message field	Description / Values
Message ID	The message ID of the PUBLISH message to be acknowledged.



# MQTT – MQ Telemetry Transport

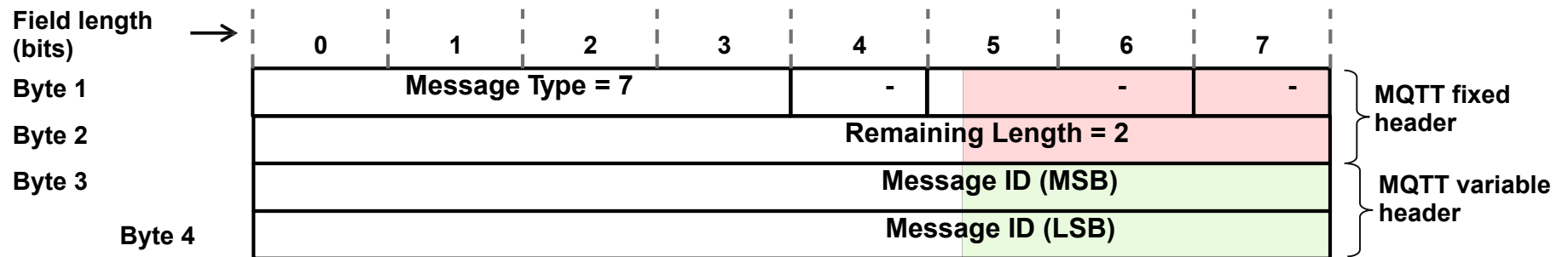
## 5. MQTT message format (10/14)

### PUBREL message format:



PUBREL message field	Description / Values
Message ID	The message ID of the PUBLISH message to be acknowledged.

### PUBCOMP message format:

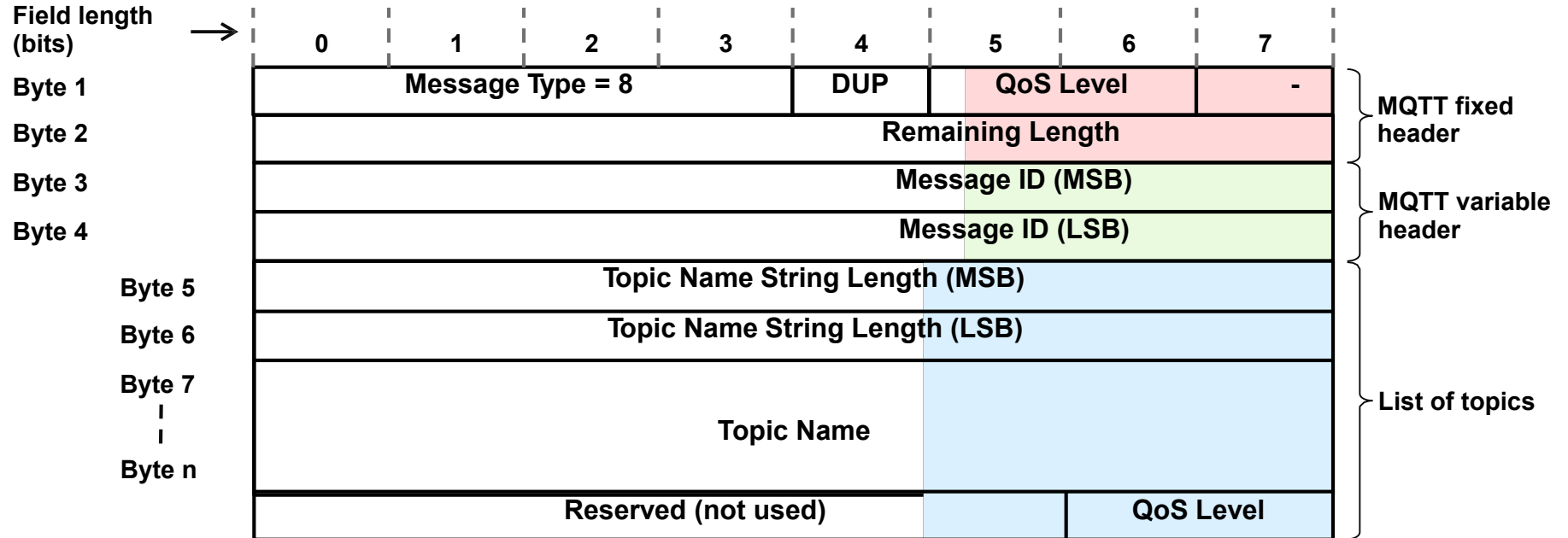


PUBCOMP message field	Description / Values
Message ID	The message ID of the PUBLISH message to be acknowledged.

# MQTT – MQ Telemetry Transport

## 5. MQTT message format (11/14)

### SUBSCRIBE message format:

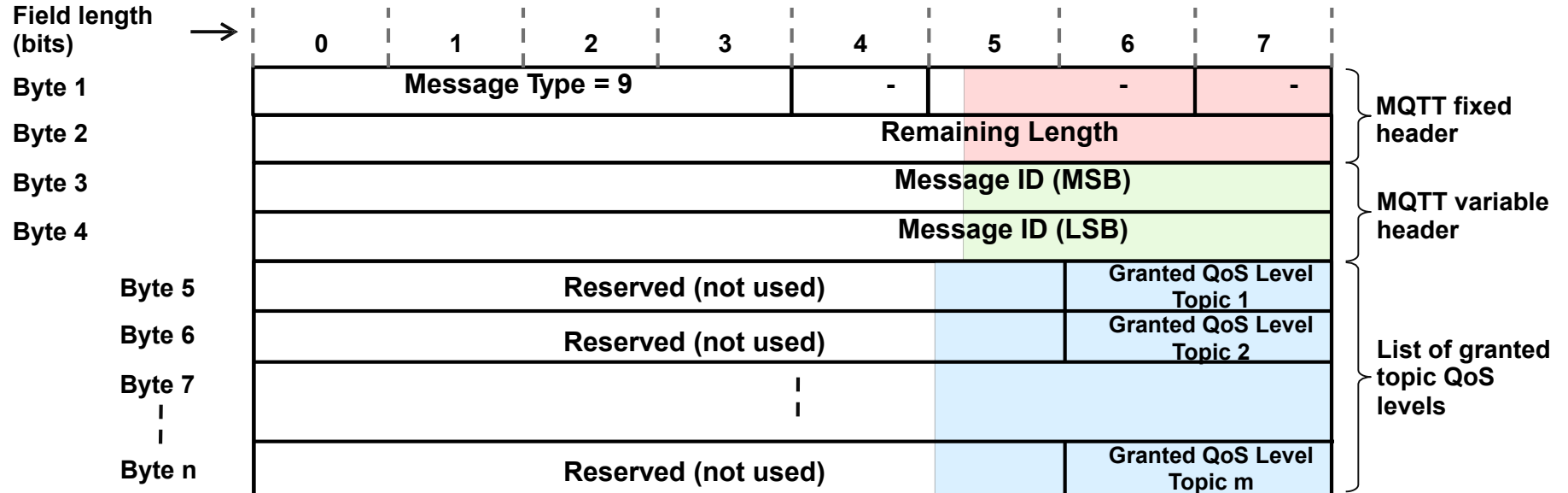


SUBSCRIBE message field	Description / Values
Message ID	The message ID field is used for acknowledgment of the SUBSCRIBE message since these have a QoS level of 1.
Topic Name with Topic Name String Length	Name of topic to which the client subscribes. The first 2 bytes of the topic name field indicate the topic name string length. Topic name strings can contain wildcard characters as explained under <a href="#">Topic wildcards</a> . Multiple topic names along with their requested QoS level may appear in a SUBSCRIBE message.
QoS Level	QoS level at which the clients wants to receive messages from the given topic.

# MQTT – MQ Telemetry Transport

## 5. MQTT message format (12/14)

### SUBACK message format:

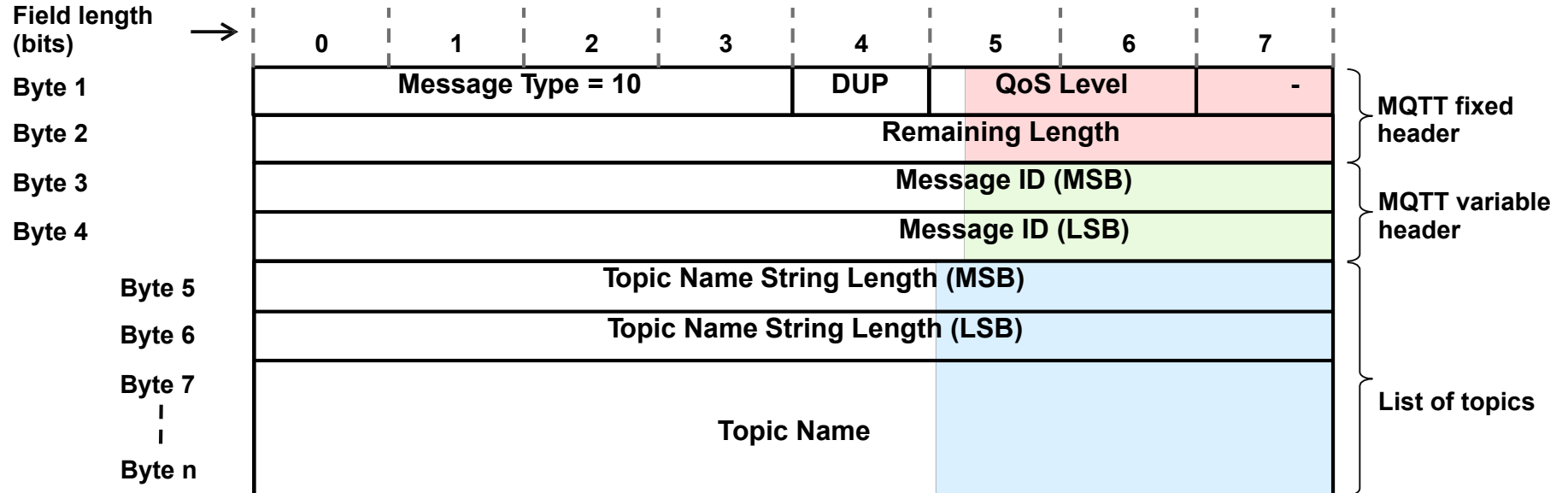


SUBACK message field	Description / Values
Message ID	Message ID of the SUBSCRIBE message to be acknowledged.
Granted QoS Level for Topic	List of granted QoS levels for the topics list from the SUBSCRIBE message.

# MQTT – MQ Telemetry Transport

## 5. MQTT message format (13/14)

### UNSUBSCRIBE message format:

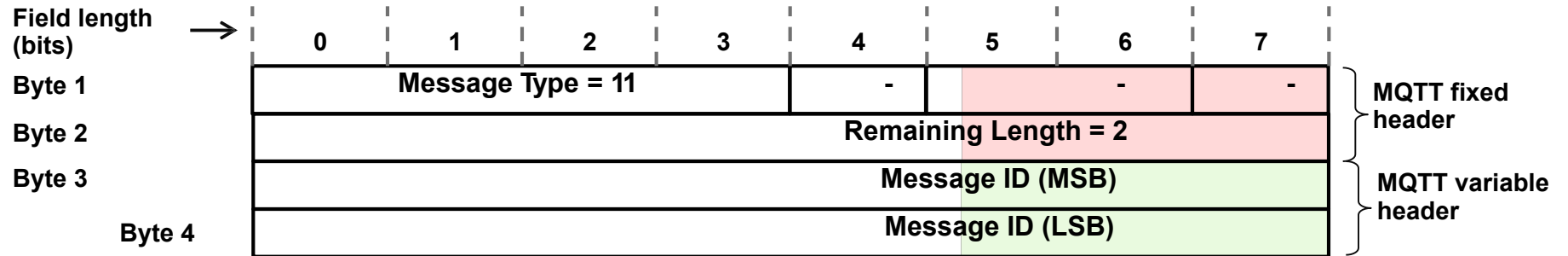


UNSUBSCRIBE message field	Description / Values
Message ID	The message ID field is used for acknowledgment of the UNSUBSCRIBE message (UNSUBSCRIBE messages have a QoS level of 1).
Topic Name with Topic Name String Length	Name of topic from which the client wants to unsubscribe. The first 2 bytes of the topic name field indicate the topic name string length. Topic name strings can contain wildcard characters as explained under <a href="#">Topic wildcards</a> . Multiple topic names may appear in an UNSUBSCRIBE message.

# MQTT – MQ Telemetry Transport

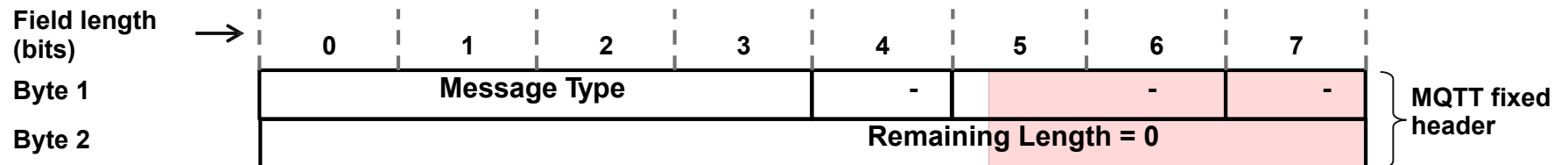
## 5. MQTT message format (14/14)

### UNSUBACK message format:



UNSUBACK message field	Description / Values
Message ID	The message ID of the UNSUBSCRIBE message to be acknowledged.

### DISCONNECT, PINGREQ, PINGRESP message formats:



# ■ MQTT Demo

# Questions & Answers

