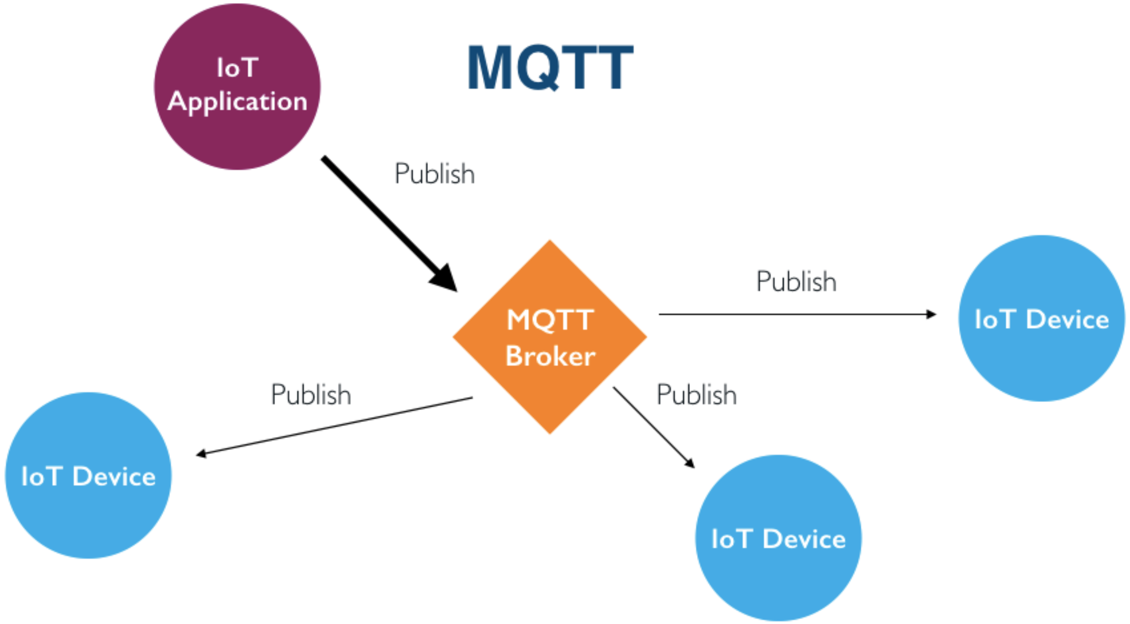


IoT Management Platform based on MQTT Technology

Presenter： Huang Hsu-Hong 黃旭弘

Advisor: Wei Hung-Yu

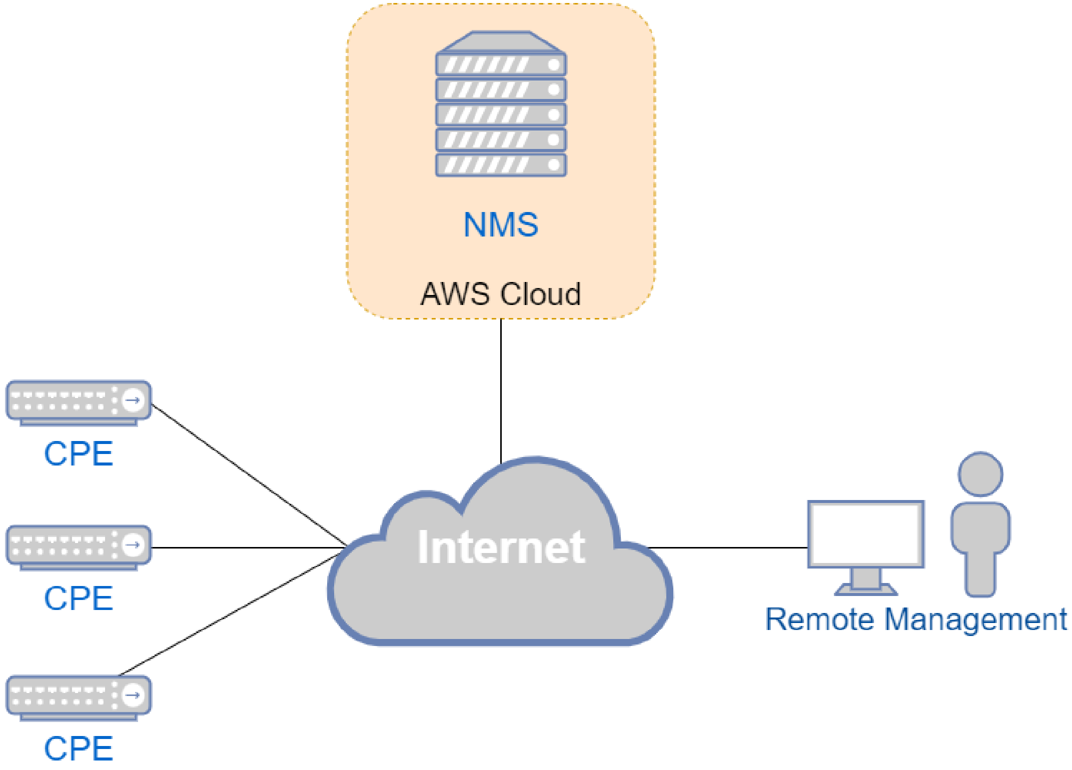
Feb. 22th 2022

Outline

- Background
- MQTT
- DEMO
- Related Works and References

Background

Network Management System



- A network management system (NMS) is an application or set of applications that **lets network engineers manage a network's** independent components inside a bigger network management framework.
- Key functions:
  - AM: Accounting Management
  - SM: Security Management
  - CM: Configuration Management
  - FM: Fault Management
  - PM: Performance Management

Motivation

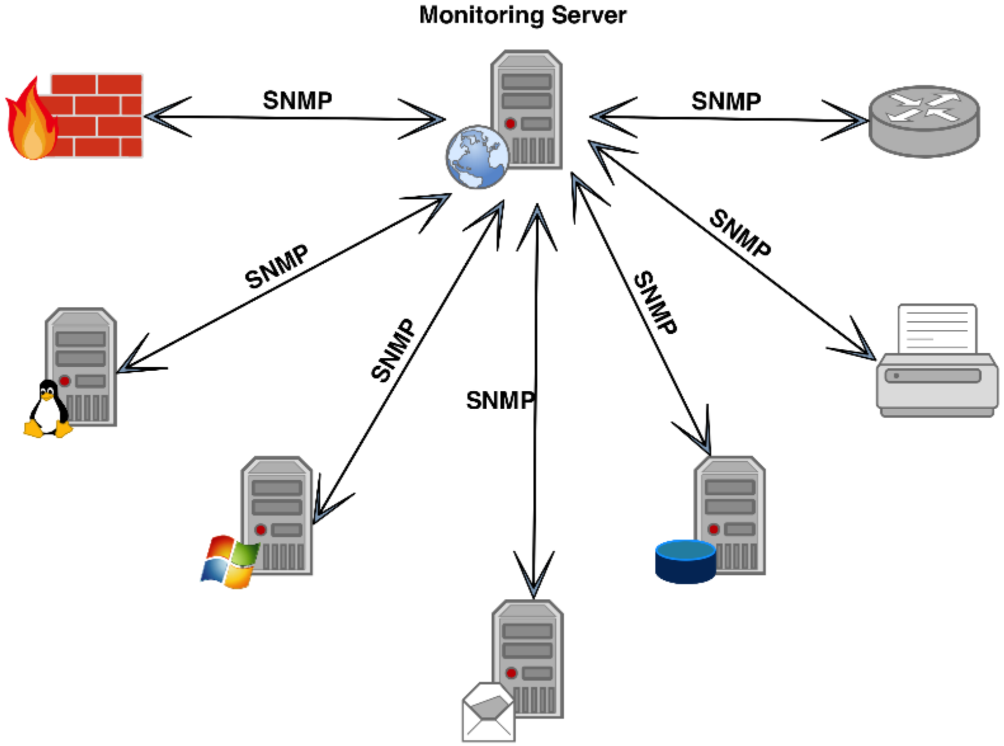
- **Device Management**
  - A. License Key Deployment
  - B. FOTA Upgrade Management
- **Remote Troubleshooting**
  - A. Device Data Monitoring & Collection(Black Box)
    - a) CPU, Memory
    - b) FW Version
    - c) Network Traffic
    - d) LTE Signal (RSRP, RSRQ, Band, CA Info)
    - e) Logs, dmesg
  - B. Remote Task Execution
    - a) Build Reverse SSH Tunnel
    - b) Diagnostic Testing
    - c) Reboot
  - C. Remote Terminal Debug Screen

Management Protocol Standardization

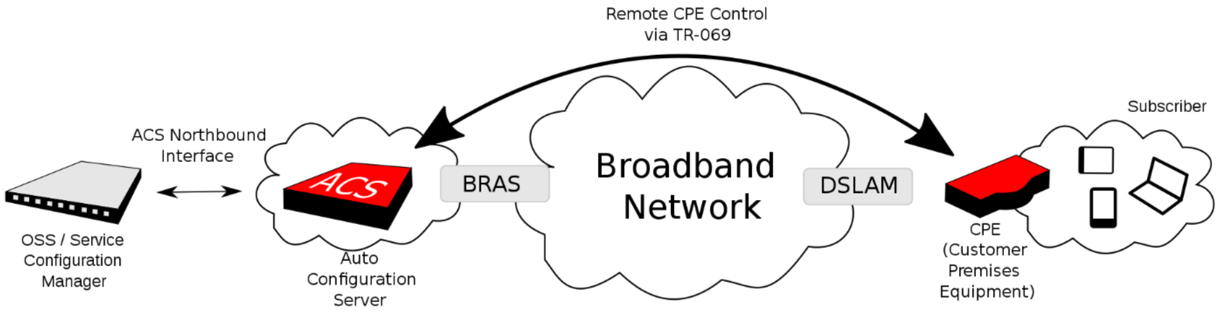
- **SNMP**
  - IETF standard (RFC 2578, 3411, 3418)
- **TR069**
  - Broadband Forum standard
- **AMQP**
  - OASIS and ISO 19464 standard (1.0)
- **MQTT**
  - OASIS standard (v5.0)

Simple Network Management Protocol(SNMP)

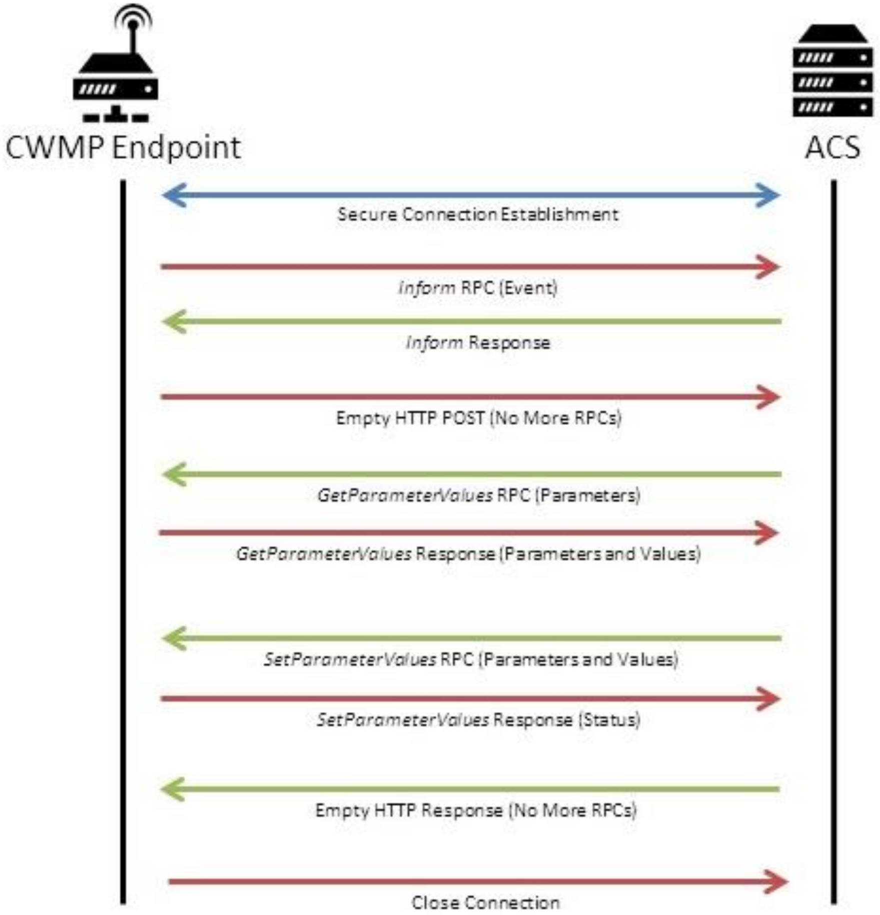
- An Internet Standard protocol for collecting and organizing information about **managed devices on IP networks**
- **Management information base** (MIB) which describe the system status and configuration.
- All SNMP messages are transported via **User Datagram Protocol (UDP)**



Technical Report 069 (TR-069)

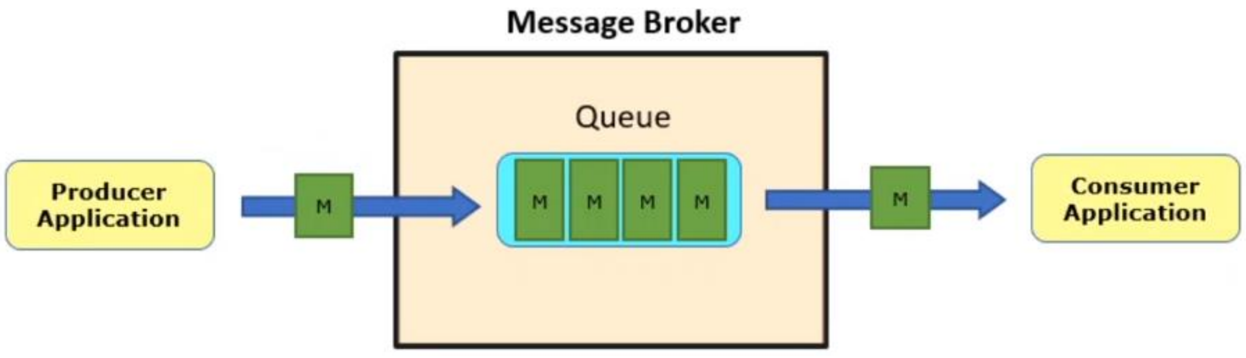


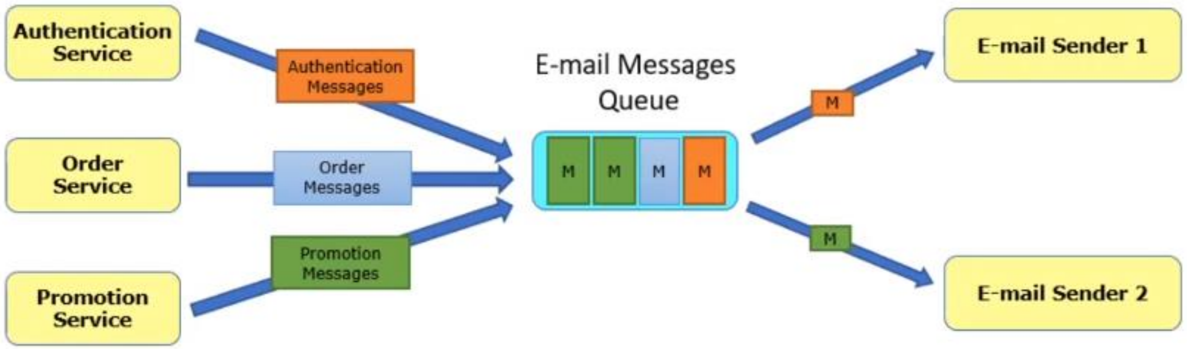
TR-069 uses the **CPE WAN Management Protocol** (**CWMP**) which provides support functions for **auto-configuration**, **software or firmware image management**, software module management, status and performance managements, and diagnostics


CWMP Endpoint
ACS
Secure Connection Establishment
Inform RPC (Event)
Inform Response
Empty HTTP POST (No More RPCs)
GetParameterValues RPC (Parameters)
GetParameterValues Response (Parameters and Values)
SetParameterValues RPC (Parameters and Values)
SetParameterValues Response (Status)
Empty HTTP Response (No More RPCs)
Close Connection

Message Queue

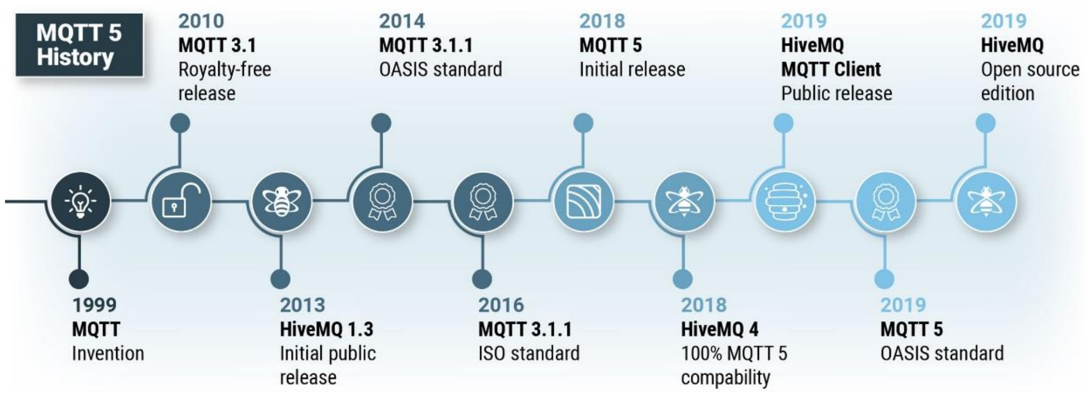
- The message queue transmits the message or content to the queue between the message **producing** and **consuming** the message, providing an **asynchronous** working protocol
- **AMQP is an open standard for communication** between applications, organizations or devices.





MQTT

MQTT History & Background



A. Full Name: MQ Telemetry Transport.

B. IBM authored the first version of the protocol in 1999.

C. Used to monitor an oil pipeline through the desert.

D. In 2013, IBM submitted MQTT v3.1 to the OASIS specification body with a charter.

E. In March, 2019, MQTT 5 took its place as the newly approved OASIS and ISO standard.

MQTT Compared to HTTP

- Both HTTP and MQTT are based on **TCP/IP**.
- HTTP uses **80/443** port, MQTT uses **1883/8883** port.
- HTTP uses Request/Response (**1 to 1**).
- MQTT uses Publish/Subscribe pattern(**1-to-1 or 1-to-many**).
- HTTP is document centric, MQTT is data centric.
- HTTP is more complex than MQTT which is simple.
- MQTT message size is smaller, with only a **2 byte** header.
- MQTT offers **3 Quality** of Service settings, with HTTP all messages get same level of service.

Structure of an MQTT Control Packet

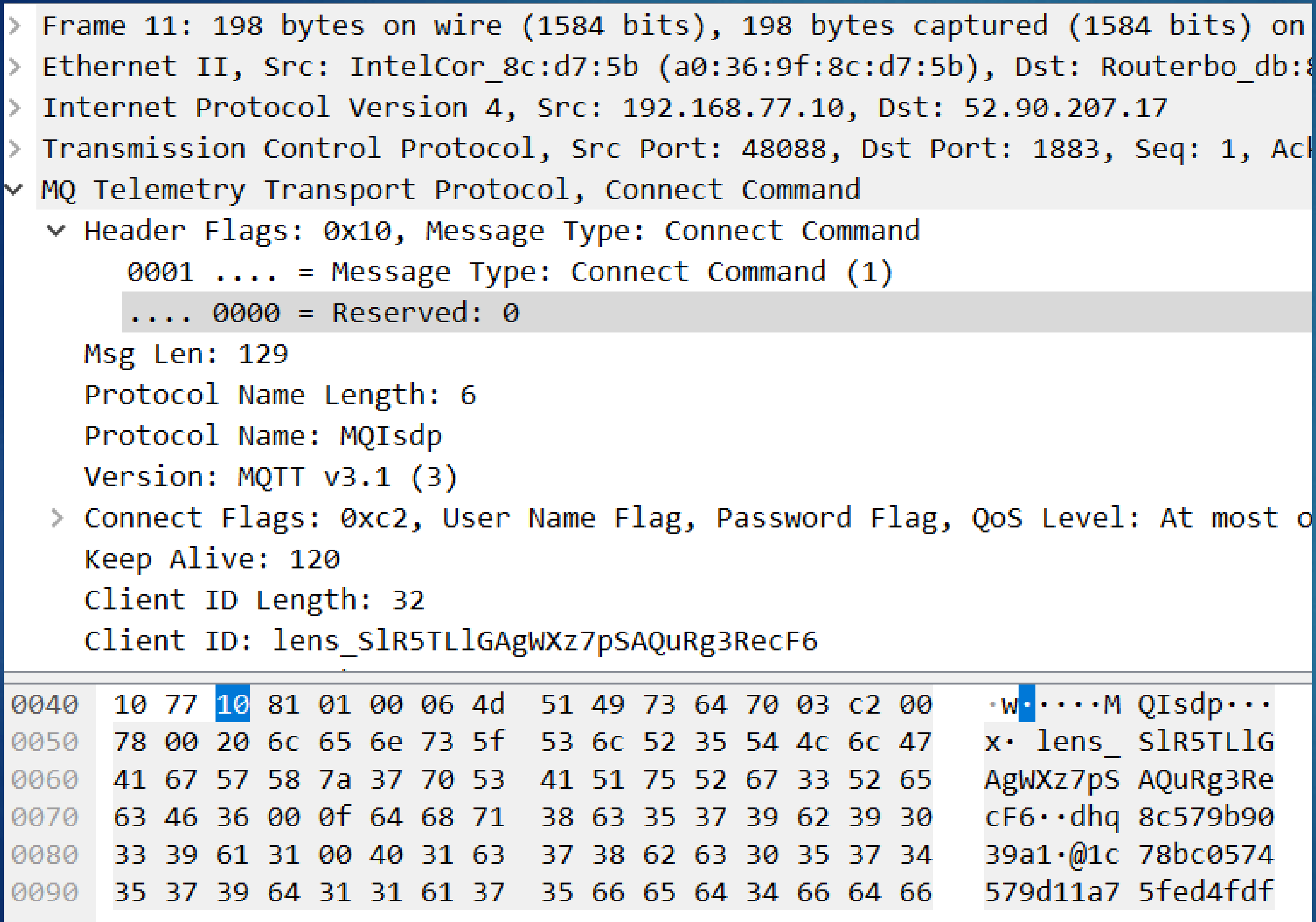
Figure 2-1 Structure of an MQTT Control Packet

Fixed Header, present in all MQTT Control Packets
Variable Header, present in some MQTT Control Packets
Payload, present in some MQTT Control Packets

Figure 2-2 Fixed Header format

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2…	Remaining Length							

MQTT Connect Package



```
> Frame 11: 198 bytes on wire (1584 bits), 198 bytes captured (1584 bits) on
> Ethernet II, Src: IntelCor_8c:d7:5b (a0:36:9f:8c:d7:5b), Dst: Routerbo_db:
> Internet Protocol Version 4, Src: 192.168.77.10, Dst: 52.90.207.17
> Transmission Control Protocol, Src Port: 48088, Dst Port: 1883, Seq: 1, Ack
v MQ Telemetry Transport Protocol, Connect Command
   v Header Flags: 0x10, Message Type: Connect Command
        0001 .... = Message Type: Connect Command (1)
        .... 0000 = Reserved: 0
     Msg Len: 129
     Protocol Name Length: 6
     Protocol Name: MQIsdp
     Version: MQTT v3.1 (3)
   > Connect Flags: 0xc2, User Name Flag, Password Flag, QoS Level: At most o
     Keep Alive: 120
     Client ID Length: 32
     Client ID: lens_SlR5TLlGAgWXz7pSAQuRg3RecF6

0040  10 77 10 81 01 00 06 4d  51 49 73 64 70 03 c2 00   ·w····M QIsdp···
0050  78 00 20 6c 65 6e 73 5f  53 6c 52 35 54 4c 6c 47   x· lens_ SlR5TLlG
0060  41 67 57 58 7a 37 70 53  41 51 75 52 67 33 52 65   AgWXz7pS AQuRg3Re
0070  63 46 36 00 0f 64 68 71  38 63 35 37 39 62 39 30   cF6··dhq 8c579b90
0080  33 39 61 31 00 40 31 63  37 38 62 63 30 35 37 34   39a1·@1c 78bc0574
0090  35 37 39 64 31 31 61 37  35 66 65 64 34 66 64 66   579d11a7 5fed4fdf
```

MQTT Controls Packets Types

- **CONNECT** – Client requests a connection to a Server.
- **CONNACK** – Acknowledge connection request.
- **PUBLISH** – Publish message.
- **PUBACK** – Publish acknowledgement: Used to response to a PUBLISH Packet with QoS level 1 and it does not contains Payload.
- **PUBREC** – Publish received: Used to response to a PUBLISH Packet with QoS 2. It is the second packet of the QoS 2 protocol exchange.
- **PUBREL** – Publish release: Used to response to a PUBREC Packet.
- **PUBCOM** – Publish complete: Use to response to a PUBREL Packet, and the final packet of the QoS 2 protocol exchange.

Table 2-1 MQTT Control Packet types

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Connection request
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment (QoS 1)
PUBREC	5	Client to Server or Server to Client	Publish received (QoS 2 delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (QoS 2 delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (QoS 2 delivery part 3)
SUBSCRIBE	8	Client to Server	Subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server or Server to Client	Disconnect notification
AUTH	15	Client to Server or Server to Client	Authentication exchange

Table 2-2 MQTT Control Flag Bits

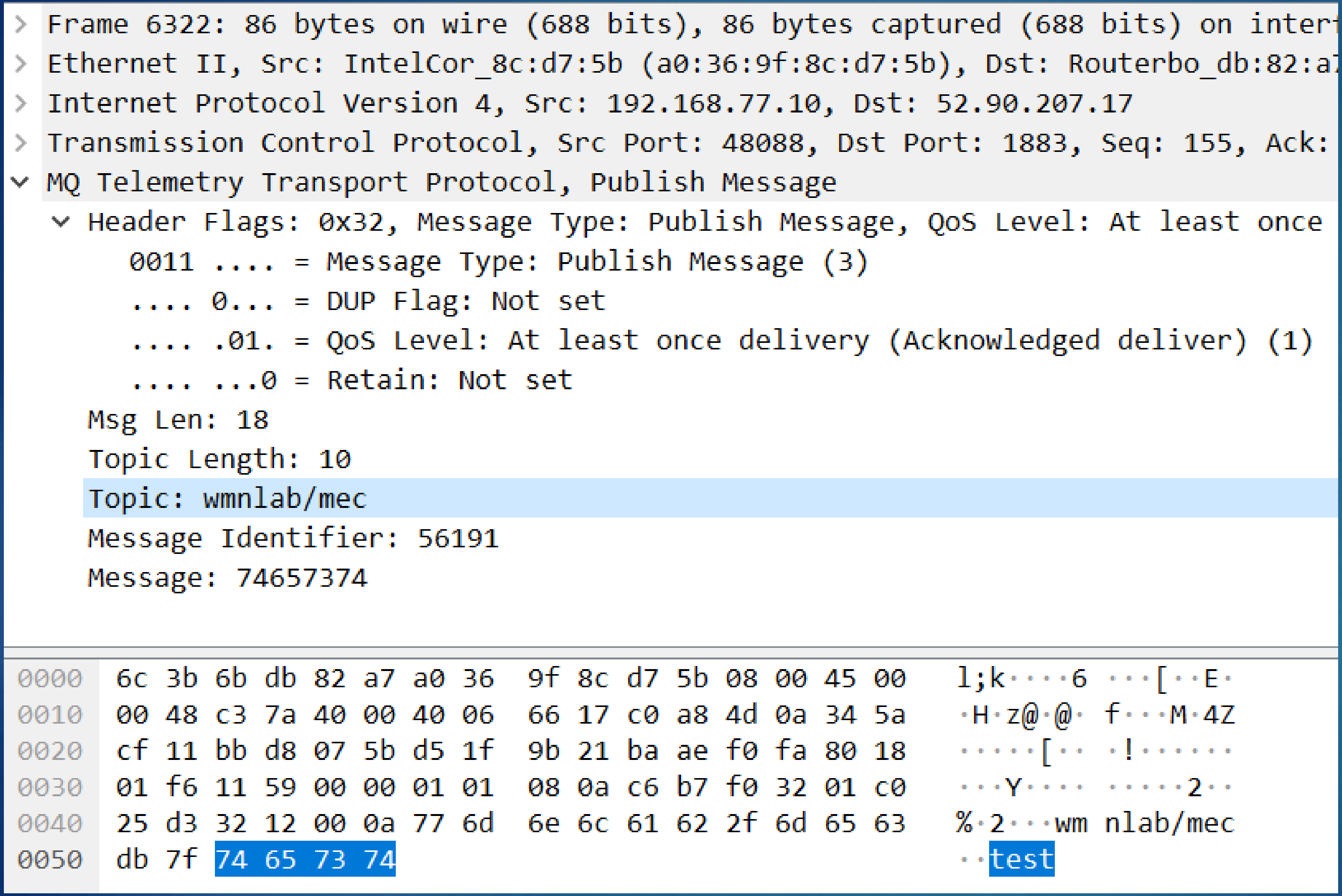
MQTT Control Packet	Fixed Header flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT v5.0	DUP	QoS		RETAIN
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0
AUTH	Reserved	0	0	0	0

DUP = Duplicate delivery of a PUBLISH packet

QoS = PUBLISH Quality of Service

RETAIN = PUBLISH retained message flag

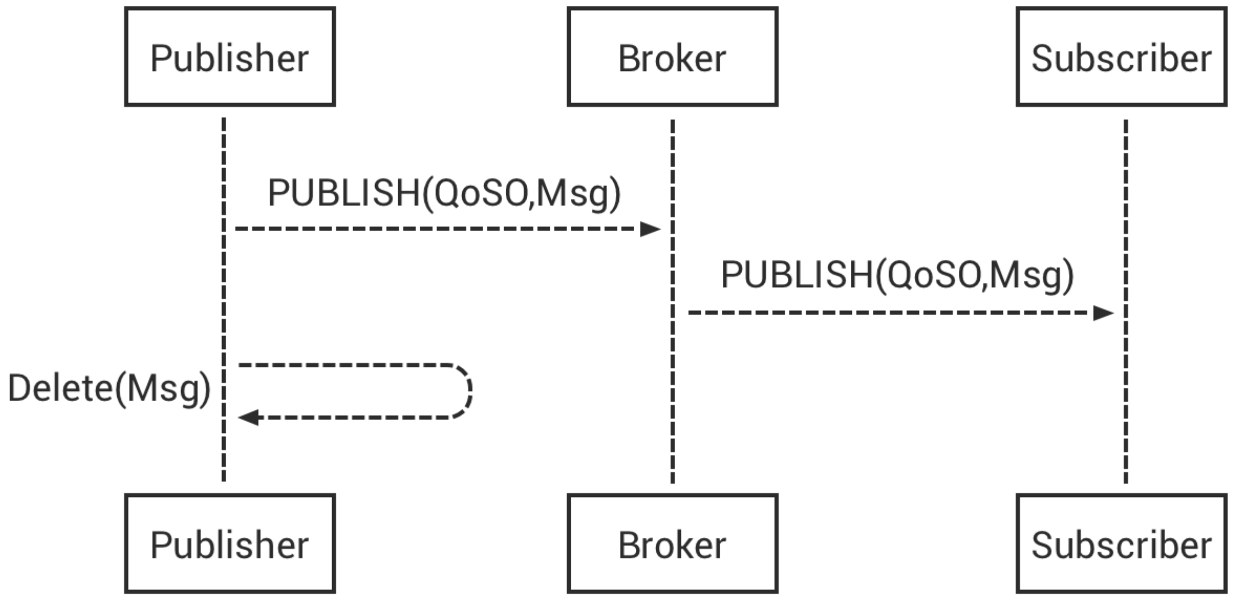
MQTT Publish Package



```
> Frame 6322: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on intert
> Ethernet II, Src: IntelCor_8c:d7:5b (a0:36:9f:8c:d7:5b), Dst: Routerbo_db:82:a
> Internet Protocol Version 4, Src: 192.168.77.10, Dst: 52.90.207.17
> Transmission Control Protocol, Src Port: 48088, Dst Port: 1883, Seq: 155, Ack:
v MQ Telemetry Transport Protocol, Publish Message
  v Header Flags: 0x32, Message Type: Publish Message, QoS Level: At least once
      0011 .... = Message Type: Publish Message (3)
      .... 0... = DUP Flag: Not set
      .... .01. = QoS Level: At least once delivery (Acknowledged deliver) (1)
      .... ...0 = Retain: Not set
    Msg Len: 18
    Topic Length: 10
    Topic: wmnlab/mec
    Message Identifier: 56191
    Message: 74657374
```

```
0000  6c 3b 6b db 82 a7 a0 36  9f 8c d7 5b 08 00 45 00   l;k····6 ···[··E·
0010  00 48 c3 7a 40 00 40 06  66 17 c0 a8 4d 0a 34 5a   ·H·z@·@· f···M·4Z
0020  cf 11 bb d8 07 5b d5 1f  9b 21 ba ae f0 fa 80 18   ·····[·· ·!······
0030  01 f6 11 59 00 00 01 01  08 0a c6 b7 f0 32 01 c0   ···Y···· ·····2··
0040  25 d3 32 12 00 0a 77 6d  6e 6c 61 62 2f 6d 65 63   %·2···wm nlab/mec
0050  db 7f 74 65 73 74                                  ··test
```

QoS 0:At most once(deliver and forgot)



Publisher

Broker

Subscriber

PUBLISH(QoSO,Msg)

PUBLISH(QoSO,Msg)

Delete(Msg)

Publisher

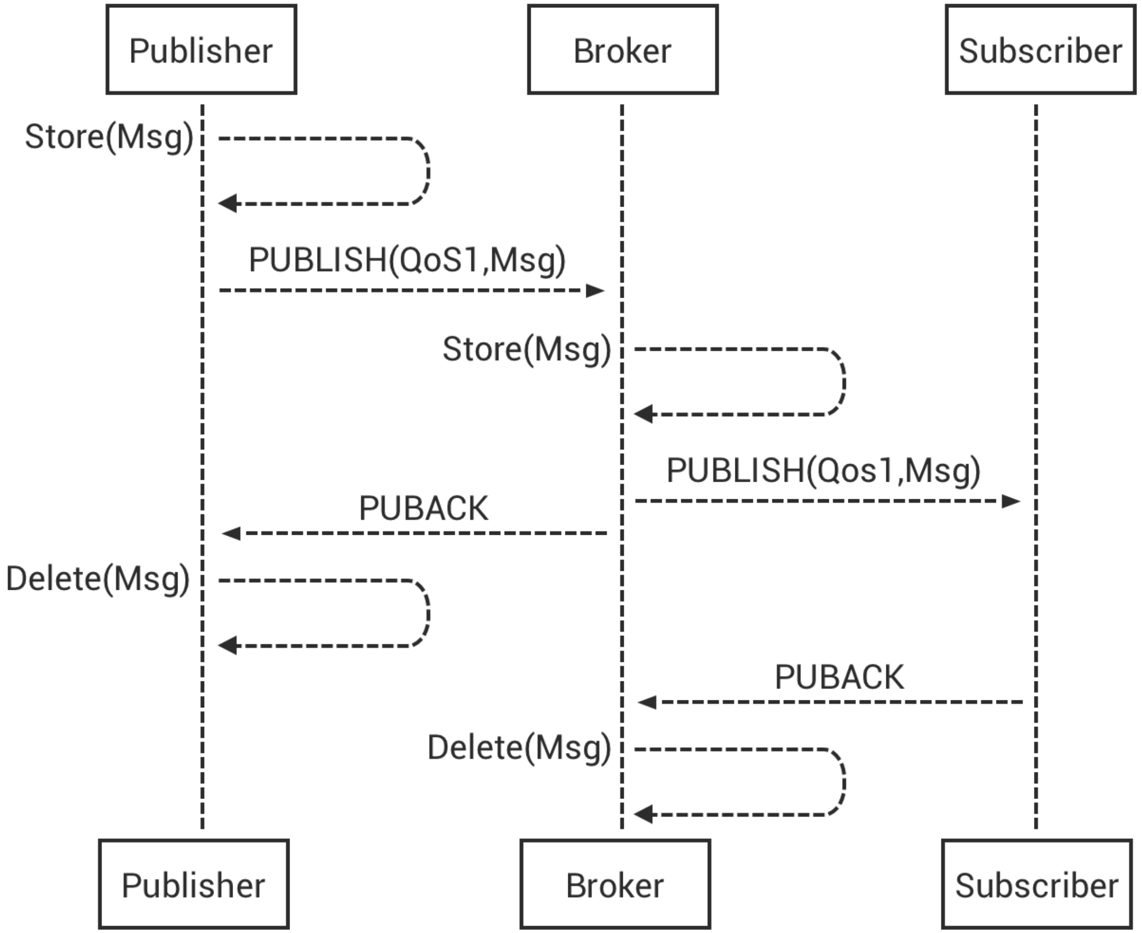
Broker

Subscriber

QoS 0
Publish once at most
(fire and forget)

QoS 1
Publish once
at least

QoS 1:At least once



Publisher

Broker

Subscriber

Store(Msg)

PUBLISH(QoS1,Msg)

Store(Msg)

PUBLISH(Qos1,Msg)

PUBACK

Delete(Msg)

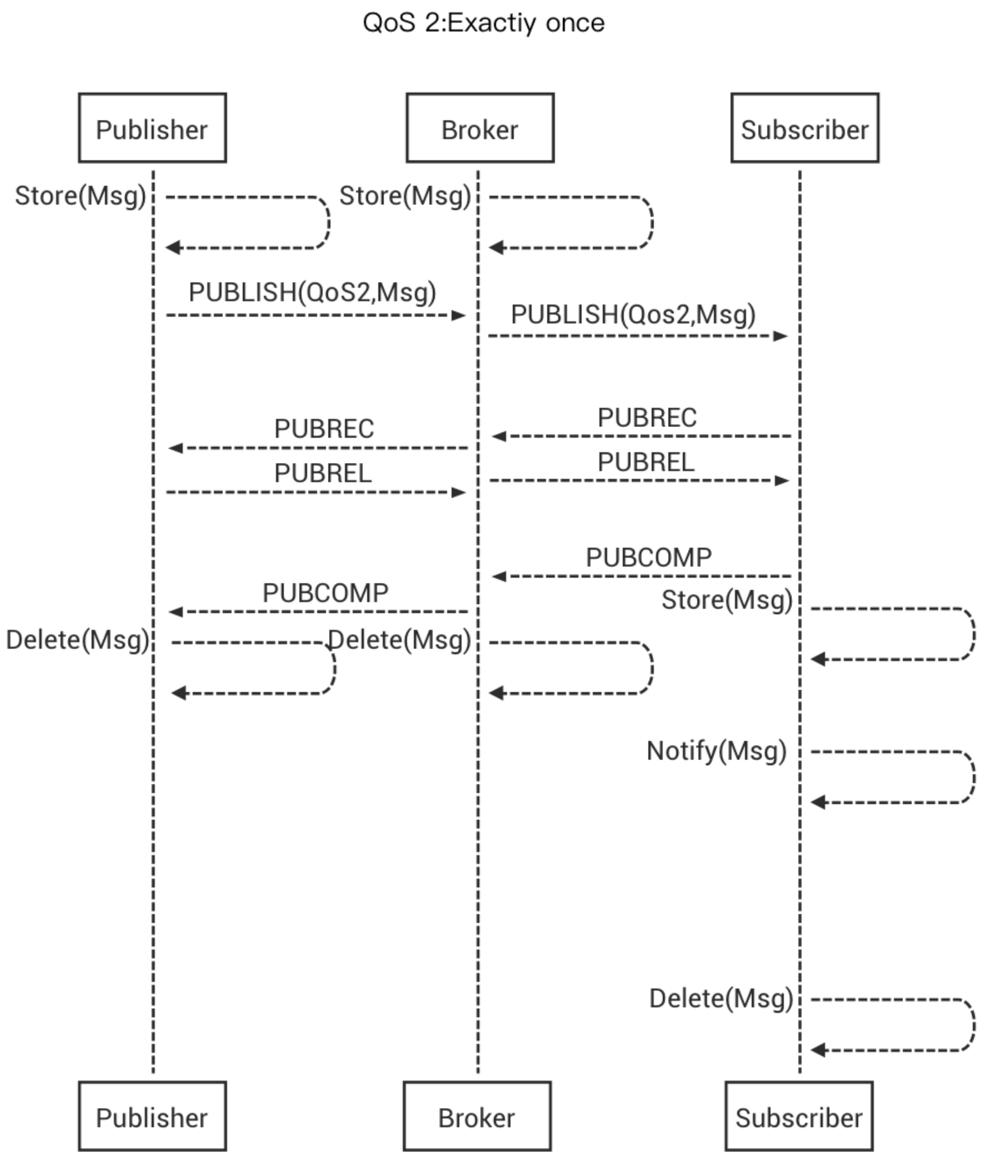
PUBACK

Delete(Msg)

Publisher

Broker

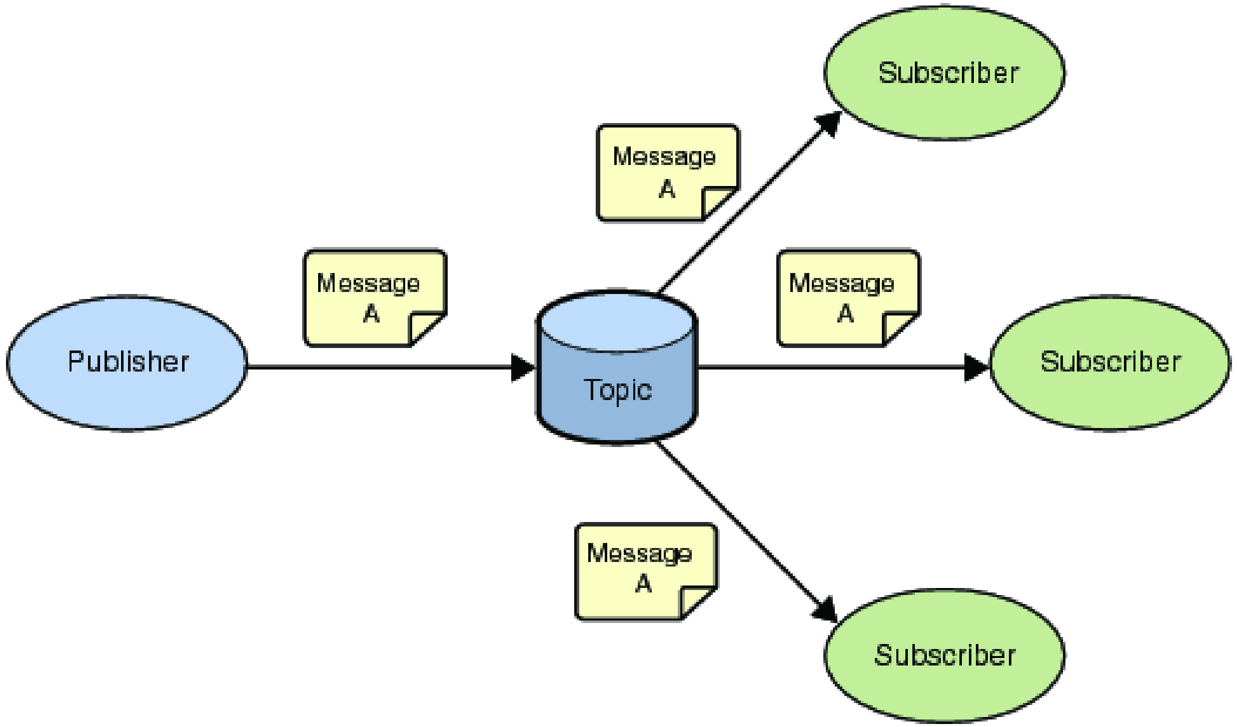
Subscriber



QoS 2
Exactly once

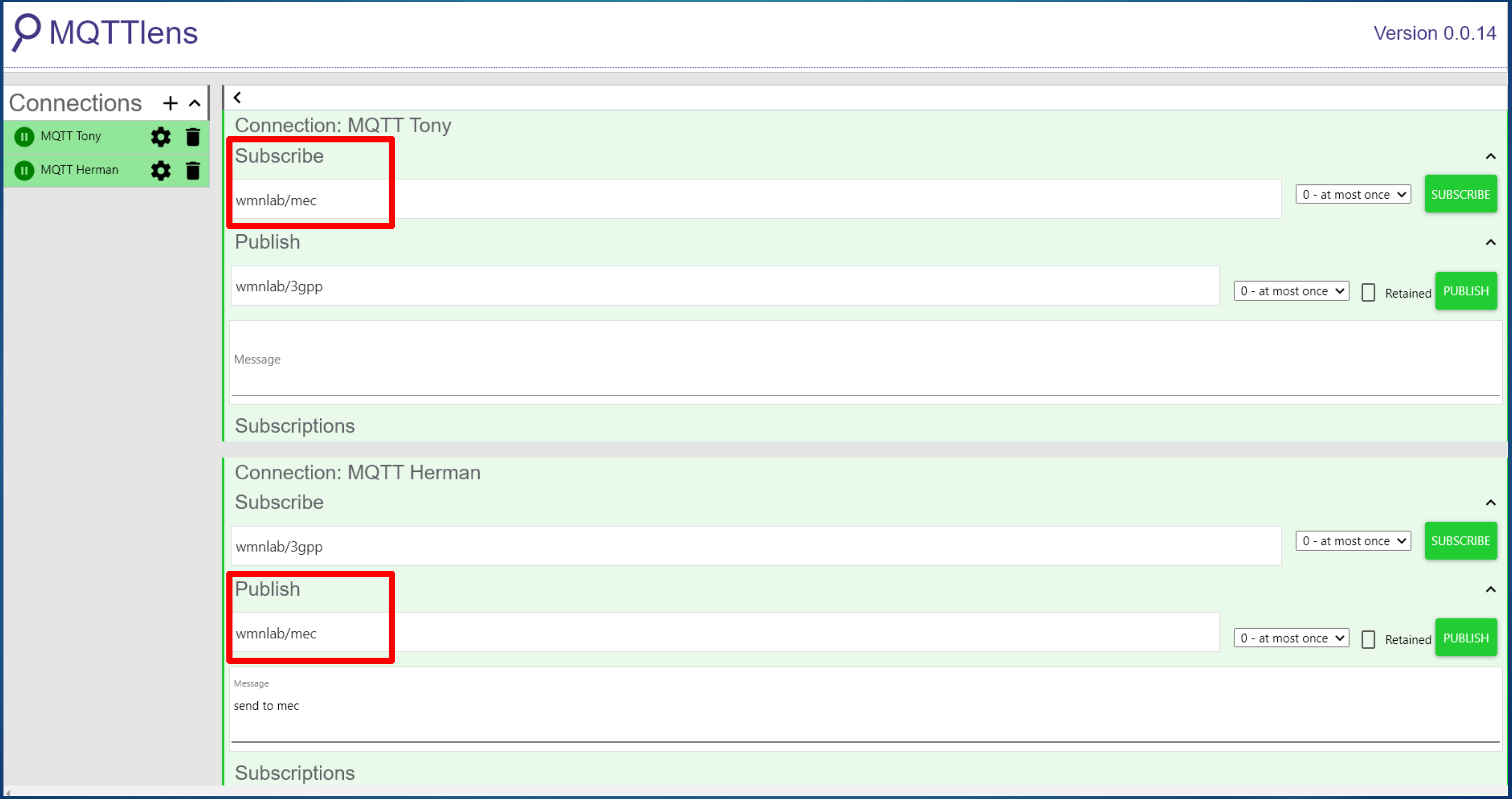
QoS 2 protocol flow diagram, non normative example

Sender Action	Control Packet	Receiver Action
Store message		
PUBLISH QoS 2, DUP 0 <Packet Identifier>		
	→	
		Method A, **Store message** or Method B, Store <Packet Identifier> then Initiate onward delivery of the Application Message[1]
		PUBREC <Packet Identifier>
	←	
Discard message, Store PUBREC received <Packet Identifier>		
PUBREL <Packet Identifier>		
	→	
		Method A, Initiate onward **delivery** of the Application **Message**[1] then **discard message** or Method B, Discard <Packet Identifier>
		Send **PUBCOMP** <Packet Identifier>
	←	
Discard stored state		



Publish/Subscribe Messaging aka One to Many

Send Message by Sub/Pub topic



Connection: MQTT Tony

Subscribe: wmnlab/mec — 0 - at most once

Publish: wmnlab/3gpp — 0 - at most once — □ Retained

Connection: MQTT Herman

Subscribe: wmnlab/3gpp — 0 - at most once

Publish: wmnlab/mec — 0 - at most once — □ Retained

Message: send to mec

A topic forms the namespace

- Is hierarchical with each “sub topic” separated by a “ / “
- An example topic space
  - A house publishes information about itself on:
    - *<country>/<region>/<town>/<postcode>/<house>/**energyConsumption***
    - *<country>/<region>/<town>/<postcode>/<house>/**solarEnergy***
    - *<country>/<region>/<town>/<postcode>/<house>/**alarmState***
    - *<country>/<region>/<town>/<postcode>/<house>/**alarmState***
  - *And subscribes for control commands:*
    - *<country>/<region>/<town>/<postcode>/<house>/thermostat/**setTemp***

Device-Oriented Topic

- A device publishes information about itself on:
  - *dwr/311-a1/00-11-22-33-44-55/hardware/loadavg*
  - *dwr/311-a1/00-11-22-33-44-55/hardware/memory*
  - *dwr/311-a1/00-11-22-33-44-55/hardware/lte/*
  - *dwr/311-a1/00-11-22-33-44-55/network/ipv4*

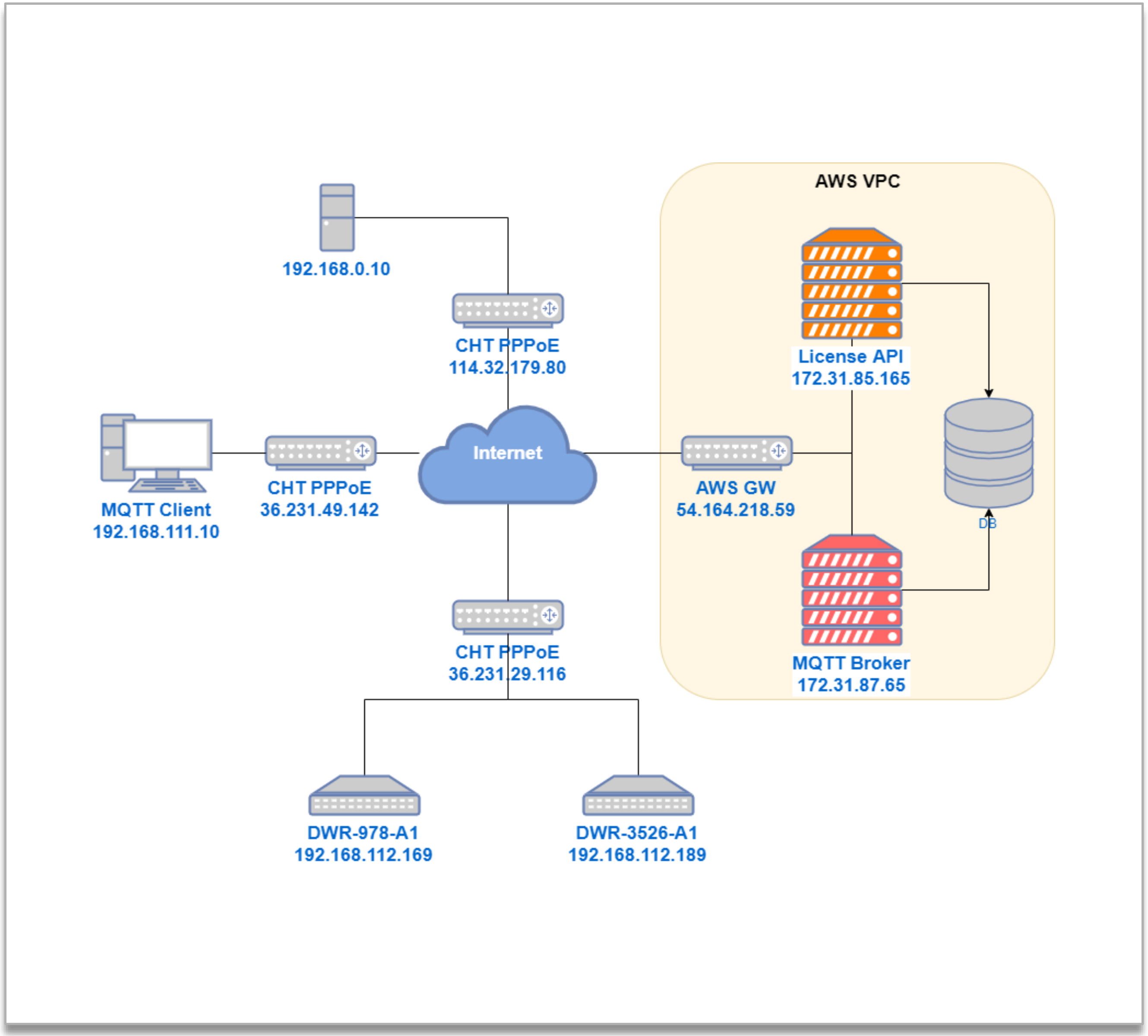
- *And subscribes for control commands:*
  - *dwr/#*
  - *dwr/311-a1/#*
  - *dwr/311-a1/00-11-22-33-44-55/#*
  - *dwr/311-a1/+/hardware/memory*

Geography-Oriented Topic

- A device publishes information about itself on:
  - *world/asia/taiwan/taipei/neihu/temperature*
  - *world/asia/taiwan/kaohsiung/sanmin/temperature*
  - *world/asia/taiwan/taipei/neihu/humidity*
  - *world/asia/taiwan/taipei/neihu/pm2dot5*
  - *world/asia/taiwan/taipei/neihu/wind*

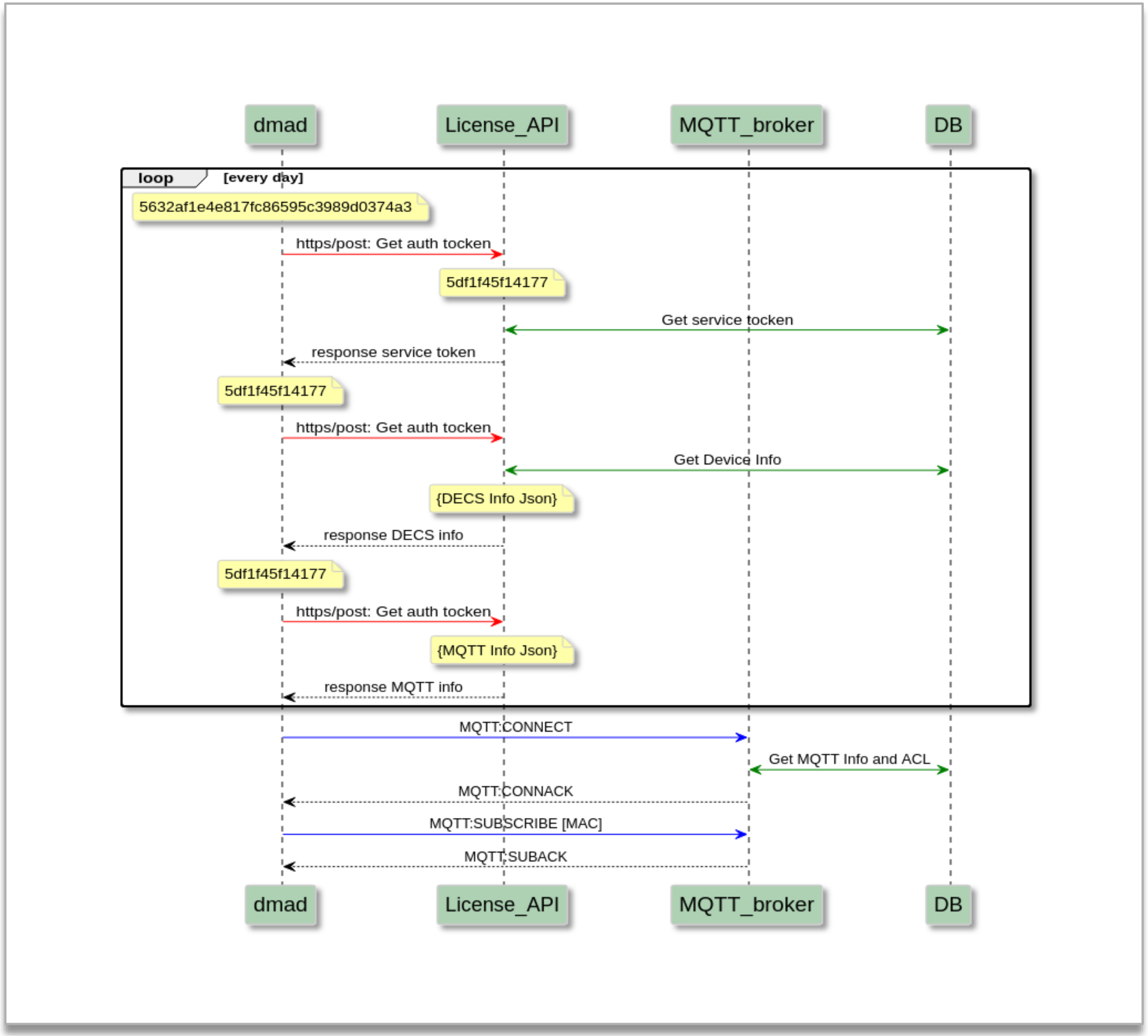
- *And subscribes for control commands:*
  - *world/asia/taiwan/#*
  - *world/asia/taiwan/taipei/+/temperature*
  - *world/asia/taiwan/+/+/temperature*

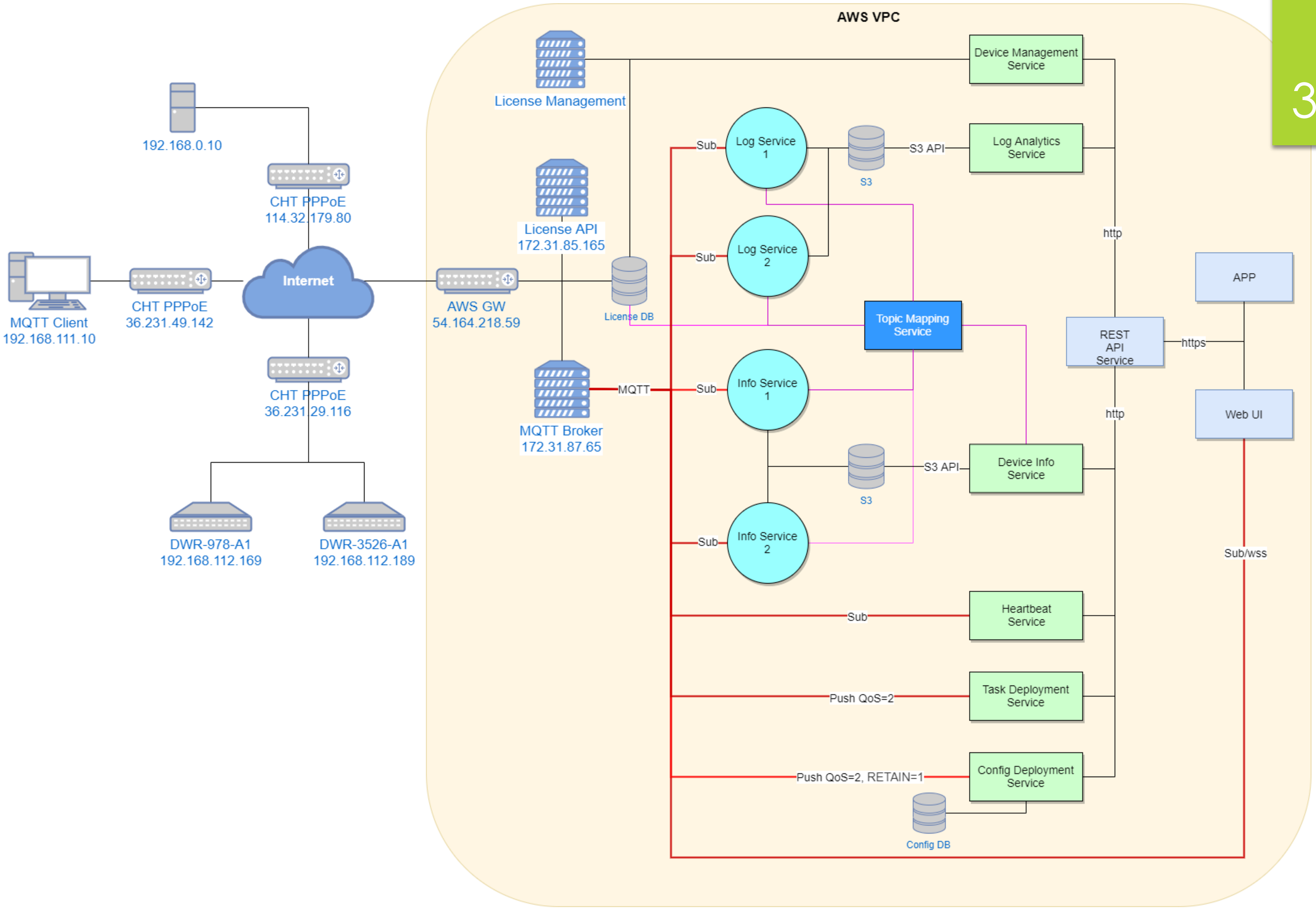
DEMO



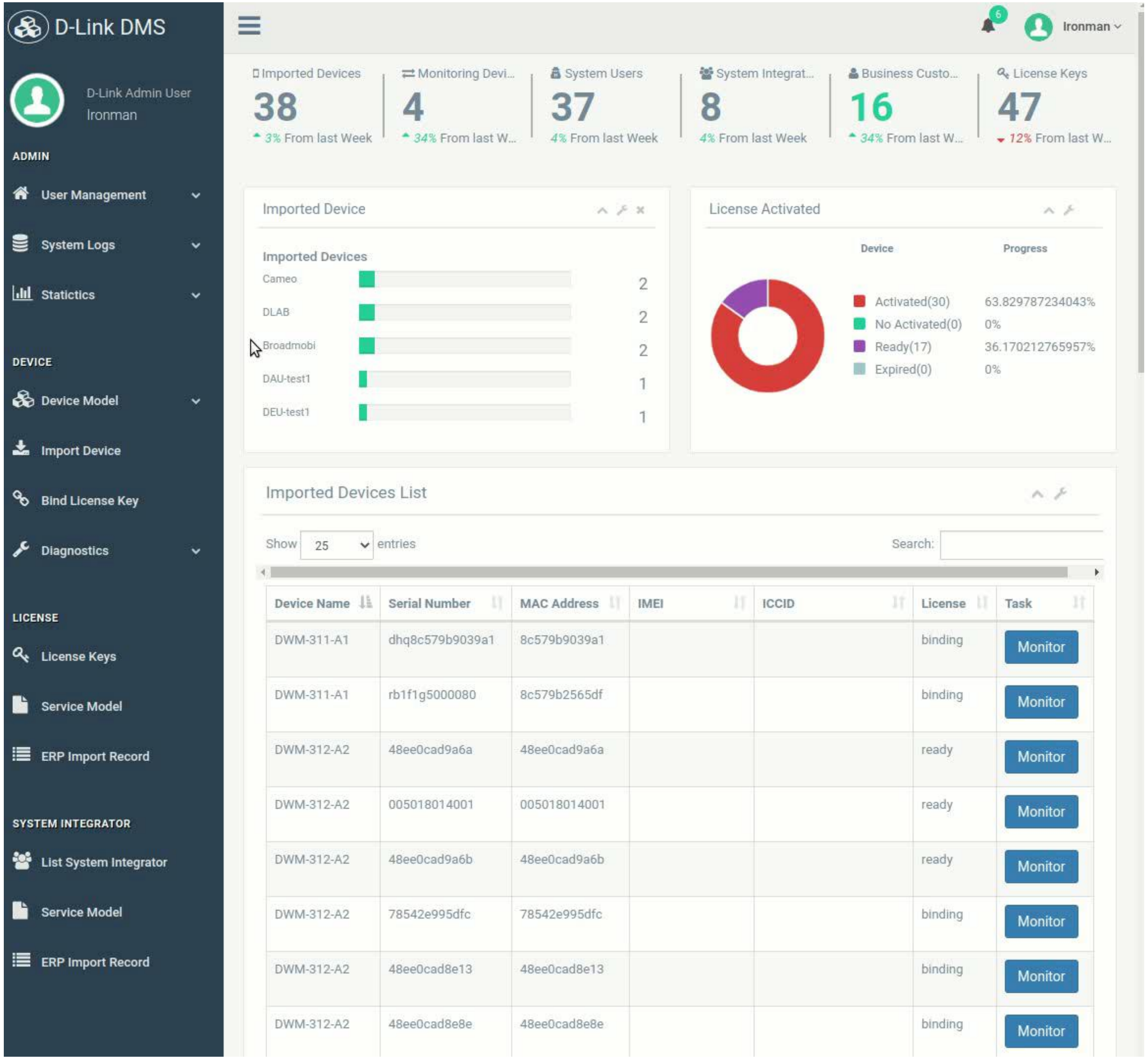
Network Management System

Connecting Procedure




AWS VPC
192.168.0.10
CHT PPPoE
114.32.179.80
MQTT Client
192.168.111.10
CHT PPPoE
36.231.49.142
Internet
CHT PPPoE
36.231.29.116
DWR-978-A1
192.168.112.169
DWR-3526-A1
192.168.112.189
AWS GW
54.164.218.59
License Management
License API
172.31.85.165
License DB
MQTT Broker
172.31.87.65
MQTT
Device Management Service
Sub
Log Service 1
S3
S3 API
Log Analytics Service
Sub
Log Service 2
http
Topic Mapping Service
REST API Service
https
APP
Sub
Info Service 1
http
Web UI
S3
S3 API
Device Info Service
Sub
Info Service 2
Sub/wss
Sub
Heartbeat Service
Push QoS=2
Task Deployment Service
Push QoS=2, RETAIN=1
Config Deployment Service
Config DB

IoT Management Platform



D-Link DMS

D-Link Admin User
Ironman

Ironman

ADMIN
- User Management
- System Logs
- Statictics

DEVICE
- Device Model
- Import Device
- Bind License Key
- Diagnostics

LICENSE
- License Keys
- Service Model
- ERP Import Record

SYSTEM INTEGRATOR
- List System Integrator
- Service Model
- ERP Import Record

Imported Devices	Monitoring Devi...	System Users	System Integrat...	Business Custo...	License Keys
38	4	37	8	16	47
3% From last Week	34% From last W...	4% From last Week	4% From last Week	34% From last W...	12% From last W...

Imported Device

Imported Devices

Cameo	2
DLAB	2
Broadmobi	2
DAU-test1	1
DEU-test1	1

License Activated

Device	Progress
🟥 Activated(30)	63.829787234043%
🟩 No Activated(0)	0%
🟪 Ready(17)	36.170212765957%
■ Expired(0)	0%

Imported Devices List

Show 25 entries

Search:

Device Name	Serial Number	MAC Address	IMEI	ICCID	License	Task
DWM-311-A1	dhq8c579b9039a1	8c579b9039a1			binding	Monitor
DWM-311-A1	rb1f1g5000080	8c579b2565df			binding	Monitor
DWM-312-A2	48ee0cad9a6a	48ee0cad9a6a			ready	Monitor
DWM-312-A2	005018014001	005018014001			ready	Monitor
DWM-312-A2	48ee0cad9a6b	48ee0cad9a6b			ready	Monitor
DWM-312-A2	78542e995dfc	78542e995dfc			binding	Monitor
DWM-312-A2	48ee0cad8e13	48ee0cad8e13			binding	Monitor
DWM-312-A2	48ee0cad8e8e	48ee0cad8e8e			binding	Monitor

Related Works and References

MQTT Brokers

Implementation	Developed by	Open source	license	Copyright	Written in	Type	Latest stable release	Origin
EMQ X	EMQ	Yes	Apache License version 2.0	EMQ Enterprise, Inc.	Erlang	Broker	V3.0, 2019-04-03	China
HiveMQ Community Edition	dc-square GmbH	Yes	Apache License version 2.0	dc-square GmbH	Java	Broker	2019.1, 2019-04-16	Germany
moquette	Andrea Selva	Yes	Apache License version 2.0	Andrea Selva	Java	Broker	0.12.1, 2019-03-03	Italy
Mosquitto	Eclipse	Yes	Eclipse Public License 1.0,	Eclipse	C	Client and Broker	1.6.3, 2019-06-18	UK
VerneMQ	VerneMQ/Erlio	Yes	Apache License version 2.0	VerneMQ/Erlio	Erlang/OTP	Broker	1.9.1 2019-08-12	Switzerland

Related Work

- Z. Y. Thean, V. Voon Yap and P. C. Teh, "Container-based MQTT Broker Cluster for Edge Computing," 2019 4th International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE), 2019, pp. 1-6, doi: 10.1109/ICRAIE47735.2019.9037775.

- T. Rausch, S. Nastic and S. Dustdar, "EMMA: Distributed QoS-Aware MQTT Middleware for Edge Computing Applications," *2018 IEEE International Conference on Cloud Engineering (IC2E)*, 2018, pp. 191-197, doi: 10.1109/IC2E.2018.00043.

- S. Pešić, M. Radovanović and M. Ivanović, "An MQTT-based Resource Management Framework for Edge Computing Systems," *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 2020, pp. 1-7, doi: 10.1109/INISTA49547.2020.9194690.

Reference

- MQTT Tools
  - MQTT Lens https://chrome.google.com/webstore/detail/mqttlens/hemojaaeigabkbcookmlgmdigohjobjm?hl=zh-TW
  - ThingSpeak https://thingspeak.com/channels
- MQTT Platform
  - https://ithelp.ithome.com.tw/articles/10222255
  - EMQ X https://docs.emqx.io/broker/v3/cn/
  - HiveMQ MQTT WebSocket client
  - mosquitto-cluster https://github.com/hui6075/mosquitto-cluster
- MQTT Specs: http://docs.oasis-open.org/mqtt/mqtt/
- IoT Platform
  - https://3smarket-info.blogspot.com/2019/11/gateway.html
  - Mainflux https://github.com/mainflux/mainflux/