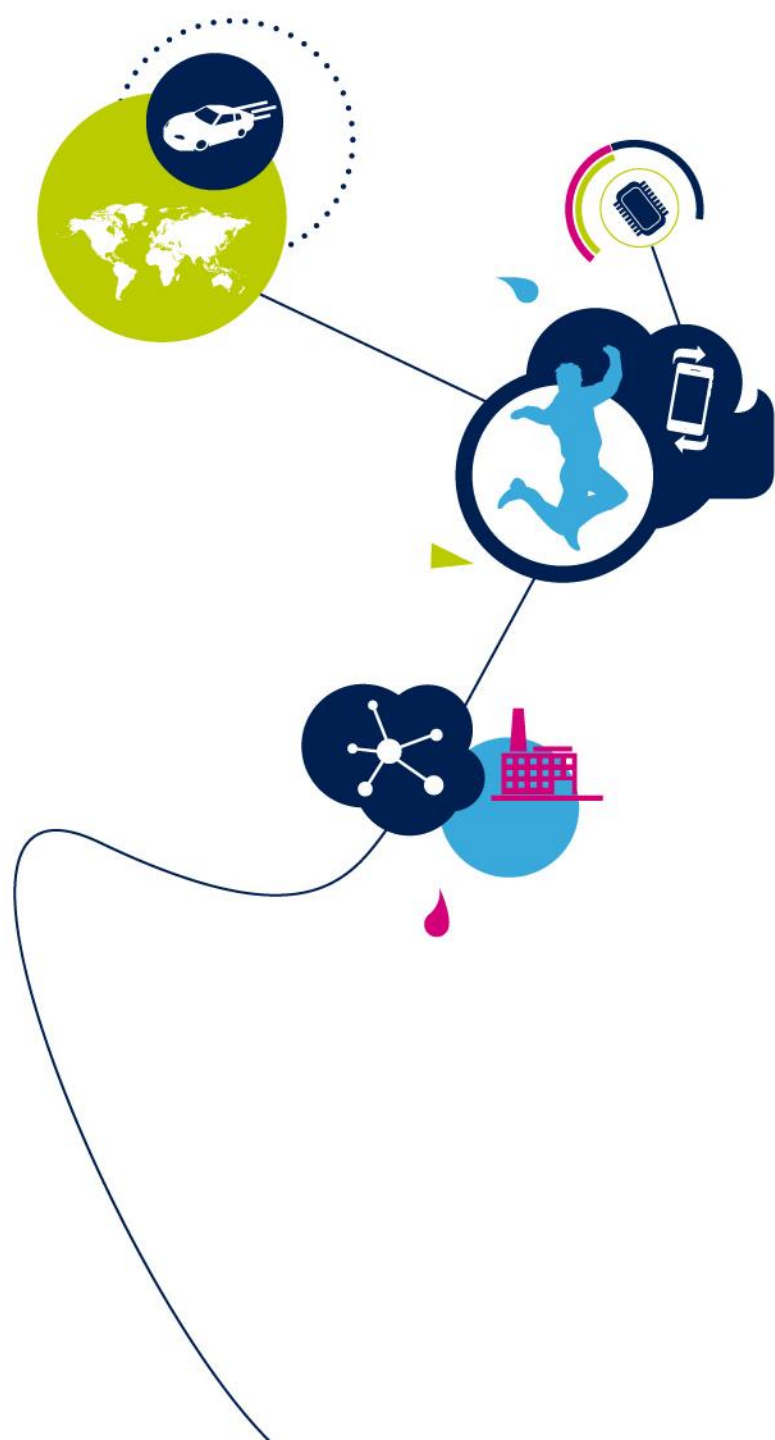


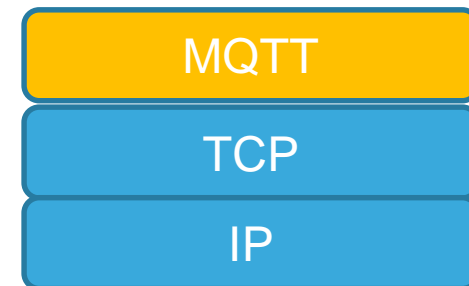
物联网通信协议





MQTT协议

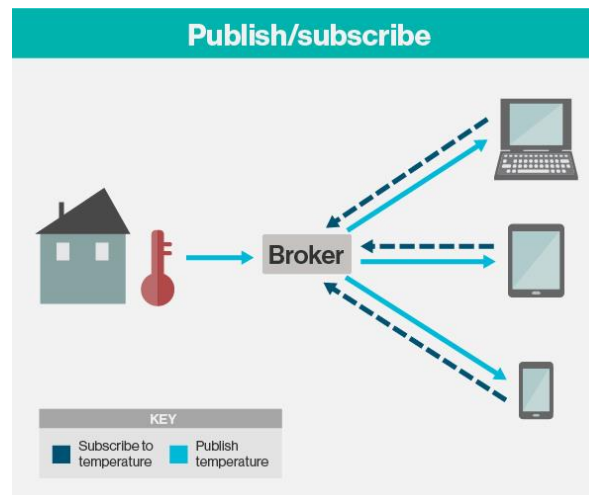
- MQTT(Message Queuing Telemetry Transport,消息队列遥测传输)
- 轻量级通信协议
- 适用资源受限设备，低带宽，高延时，不稳定网络中进行消息传输
- 运行在TCP/IP协议之上
- 客户端/服务器模式
- 发布/订阅消息模式，提供一对多消息分发
- 对传输消息有三种服务质量（QoS）
- 通知机制，在异常中断时通知相关方



MQTT怎么工作

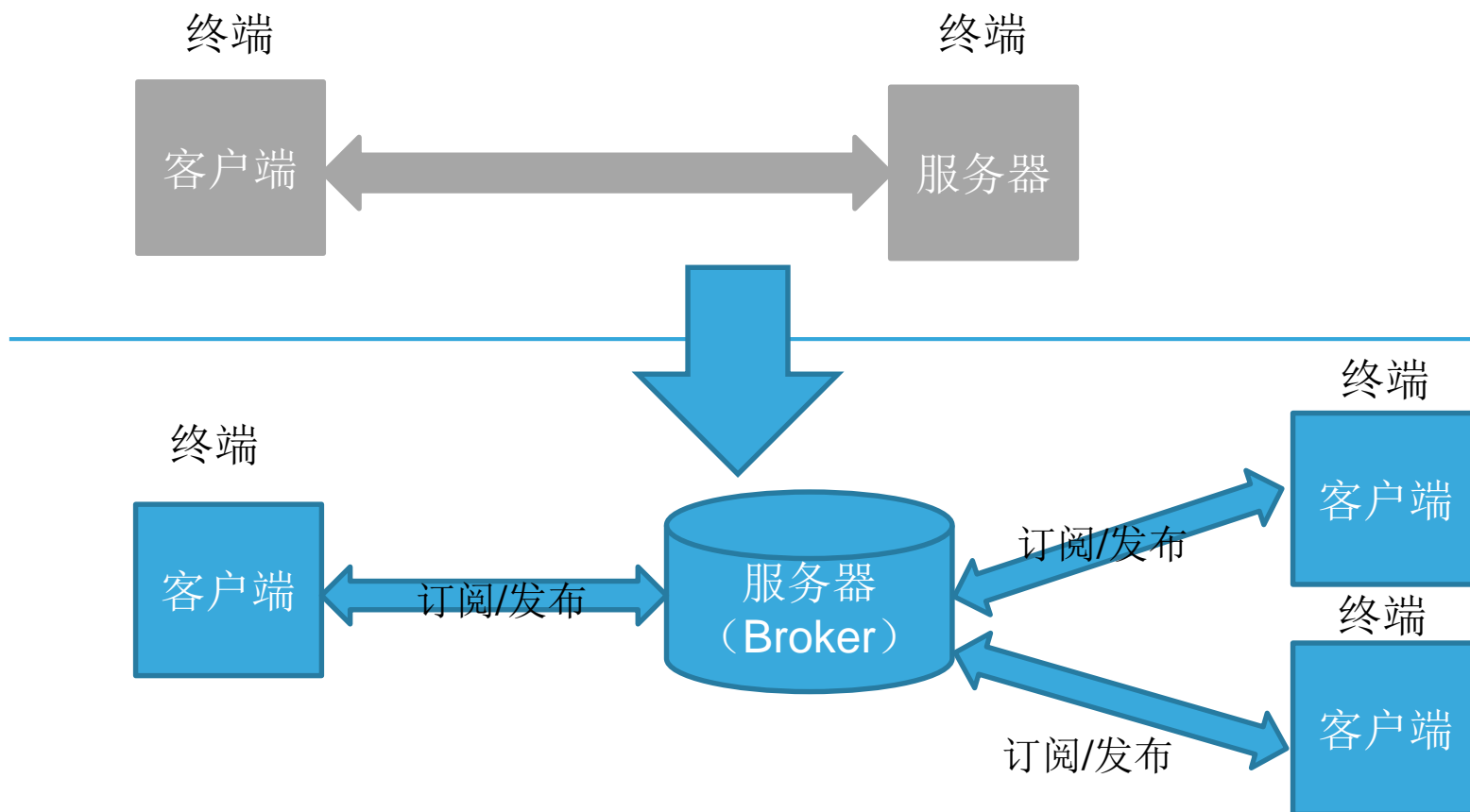
4

- 三个角色: **Broker(Server)**, 发布消息客户端, 订阅消息客户端
- 发布消息的客户端可以同时是订阅消息的客户端
- 客户端与客户端之间通过**Broker**交换消息
- 客户端与客户端之间的消息传输按主题（Topic）进行



MQTT发布/订阅模式

5



- 空间上去耦合
 - 消息发布者和消息接收者不需要知道对方的IP地址，端口
- 时间上去耦合
 - 发布者和接受者不需要同时在线

- 发布者和订阅者必须事先知道正确的topic
- 发布者并不能知道是否有节点收到消息

- 客户端
 - 消息发布者和消息接收者都是客户端
 - 总是由客户端发起到服务器的连接或断开连接
- 服务器（**Broker**）
 - 接收客户端发起的网络连接
 - 对客户端的认证和授权管理
 - 接收客户端发送的消息
 - 处理来自客户端的订阅和取消订阅的请求
 - 将消息下发到订阅客户端

- Topic
 - 附加在应用消息上的标签
 - Topic由一级或多级Topic组成，每个级别之间由“/” 分开
 - Topic区分大小写
 - 服务器根据Topic对客户端发布的消息进行管理，并分发给订阅了该Topic的客户端
- Topic Filter
 - 包含在订阅操作中，以表明客户端所感兴趣的Topic。可以是一个也可以是多个。



- 单级通配符 +

myhome / groundfloor / + / temperature

single-level wildcard
↓
only one level



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

- 多级通配符 #

myhome / groundfloor / #

multi-level wildcard
↓
only at the end
multiple topic levels



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

- 系统保留Topic
以\$开头的是服务器保留的Topic

- MQTT定义了3种QoS级别：
 - QoS0(At most once)
 - QoS1(At least once)
 - QoS2(Exactly once)
- 客户端发布消息到服务器的QoS由Publish消息的QoS决定
- 服务器发送消息到已订阅消息的客户端，QoS是由客户端订阅消息时的QoS决定
- 不同客户端，不同Topic的QoS是独立的
- 对于同一个Topic, 向服务器发布消息和服务器向客户端发送消息的QoS也可以不一样
- 客户端可以根据自己的网络情况和应用逻辑选择合适的QoS

QoS0(At most once)

10

- 消息可能被接收，也可能丢失

| Sender Action | Control Packet | Receiver Action |
|----------------------|----------------|--|
| PUBLISH QoS 0, DUP=0 | | |
| | -----> | |
| | | Deliver Application Message to appropriate onward recipient(s) |

QoS1 (At least once)

11

- 发送者保存消息直到收到响应
- 如果没有在一定时间内收到响应，则重发消息，重发的消息DUP标志必须置1
- 相同的消息，Packet Identifier相同
- 消息一定能被接收到
- 可能是一次，也可能接收到重复消息

| Sender Action | Control Packet | Receiver action |
|--|----------------|--|
| Store message | | |
| Send PUBLISH QoS 1, DUP 0, <Packet Identifier> | -----> | |
| | | Initiate onward delivery of the Application Message ¹ |
| | <----- | Send PUBACK <Packet Identifier> |
| Discard message | | |

QoS2(Exactly once)

12

- 确保消息被收到，且不会被重复接收

| 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 | | |

MQTT控制报文格式(1)

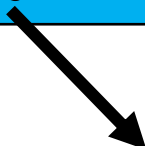
13

Fixed header, present in all MQTT Control Packets

Variable header, present in some MQTT Control Packets

Payload, present in some MQTT Control Packets

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|--|---|---|---|------------------------------------|---|---|---|
| Byte 1 | MQTT Control Packet type (定义了14中类型：连接，发布.....) | | | | Flags specific to each MQTT Packet | | | |
| Byte2 | Remaining Length | | | | | | | |



包括可变报头 (Variable header)
和负载(Payload)部分的总长度

MQTT控制报文格式(2)

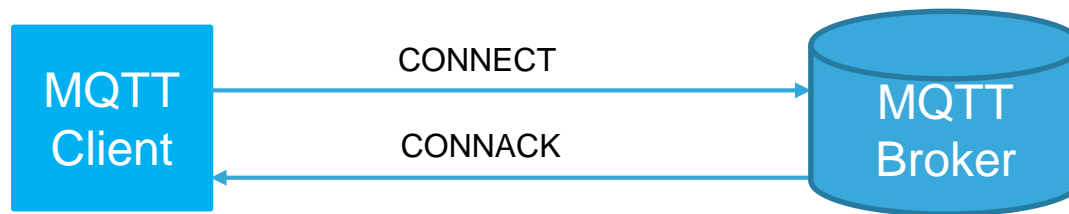
14

| |
|---|
| Fixed header, present in all MQTT Control Packets |
| Variable header, present in some MQTT Control Packets |
| Payload, present in some MQTT Control Packets |

- 不同类型的控制报文的Variable header的内容不一样
- 部分控制报文包含2字节的Packet Identifier

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|-----------------------|---|---|---|---|---|---|---|
| byte 1 | Packet Identifier MSB | | | | | | | |
| byte 2 | Packet Identifier LSB | | | | | | | |

- MQTT连接建立在TCP/IP之上
- 客户端与服务器之间建立连接，客户端之间没有直接连接
- 总是由客户端发送**CONNECT**消息来发起一次连接



- **CONNECT**消息包含的内容：
 - Client ID
 - Username,password
 - CleanSession
 - WillTopic,WillQoS, WillMessage,WillRetain
 - KeepAlive
- **CONNACK**消息包含：
 - SessionPresentFlag
 - ReturnCode

Client ---- > Server

Connect控制报文

16

| | Description | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|-----------------------------|---------------------|------------------------------|---|---|---|----------|---|---|---|-----------------|
| Byte 1 | Message Type | MQTT Control Packet type (1) | | | | Reserved | | | | Fixed Header |
| Byte 2 | Remaining Length | Remaining Length | | | | | | | | |
| Byte 3~8 (Protocol Name) | Length MSB(0) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Variable Header |
| | Length LSB(4) | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| | “M” | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | |
| | “Q” | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | |
| | “T” | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | |
| | “T” | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | |
| Byte9 | Protocol Version(4) | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| Byte10 | Connect Flag | X | X | X | X | X | X | X | X | |
| Byte11 | Keep Alive MSB | X | X | X | X | X | X | X | X | |
| Byte12 | Keep Alive LSB | X | X | X | X | X | X | X | X | |
| Byte13 | Client Identifier | | | | | | | | | Payload |
| ~ | Will Topic | | | | | | | | | |
| Byte n | Will Message | | | | | | | | | |
| | Username | | | | | | | | | |
| | Password | | | | | | | | | |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----------------|---------------|-------------|----------|---|-----------|---------------|----------|
| | User Name Flag | Password Flag | Will Retain | Will QoS | | Will Flag | Clean Session | Reserved |
| Byte10 | X | X | X | X | X | X | X | 0 |

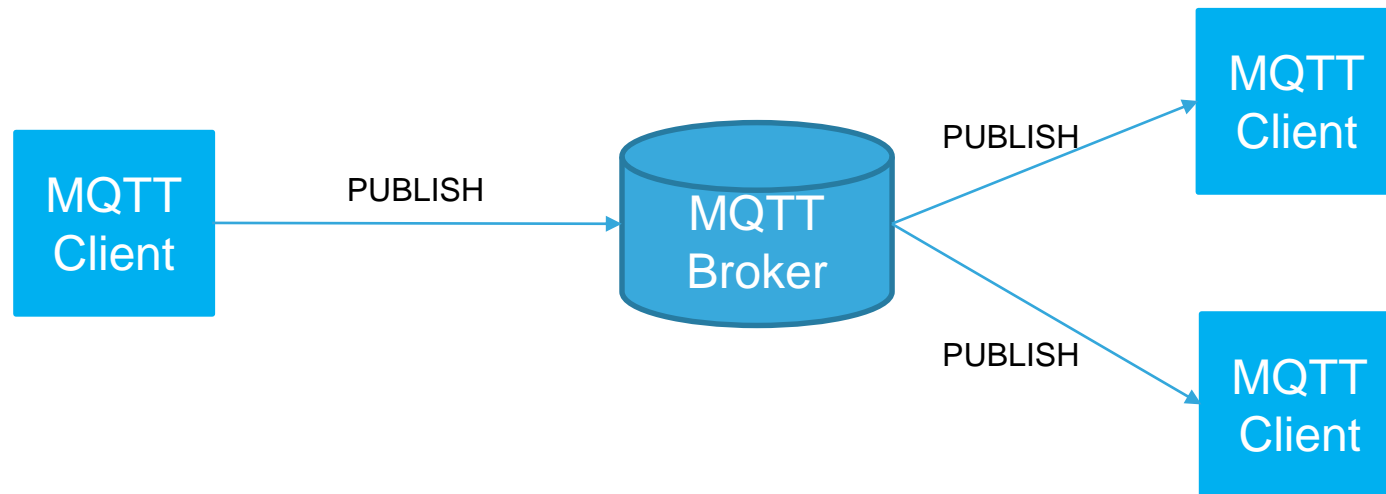
Connect Flag字节包括:

- User Name Flag, Password Flag
- Will Retain, Will QoS, Will Flag
- Clean Session

| | Description | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
|--------|---------------------------|------------------------------|---|---|---|----------|---|---|---|--------------|-----------------|
| Byte 1 | Message Type | MQTT Control Packet type (2) | | | | Reserved | | | | Fixed Header | |
| Byte 2 | Remaining Length | Remaining Length = 2 | | | | | | | | | |
| Byte 3 | Connect Acknowledge Flags | Reserved | | | | | | | | SP | Variable Header |
| Byte 4 | Connect Return Code | X | X | X | X | X | X | X | X | | |

| Value | Return Code Response | Description |
|-------|--|--|
| 0 | 0x00 Connection Accepted | Connection accepted |
| 1 | 0x01 Connection Refused, unacceptable protocol version | The Server does not support the level of the MQTT protocol requested by the Client |
| 2 | 0x02 Connection Refused, identifier rejected | The Client identifier is correct UTF-8 but not allowed by the Server |
| 3 | 0x03 Connection Refused, Server unavailable | The Network Connection has been made but the MQTT service is unavailable |
| 4 | 0x04 Connection Refused, bad user name or password | The data in the user name or password is malformed |
| 5 | 0x05 Connection Refused, not authorized | The Client is not authorized to connect |
| 6-255 | | Reserved for future use |

- PUBLISH消息包含：
 - Packet ID
 - TopicName
 - QoS
 - RetainFlag
 - Payload
 - DupFlag



Client --- > Server

Server --- > Client

Publish 控制报文

20

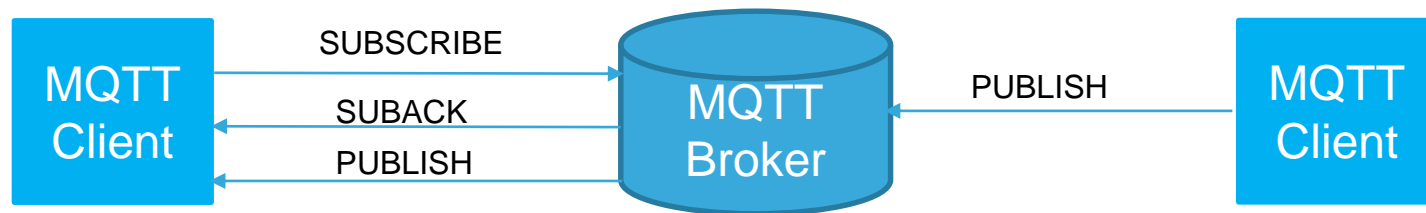
| | Description | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|--------------------|---------------------|------------------------------|---|---|---|----------|-----------|---|--------|-----------------|
| Byte 1 | Message Type | MQTT Control Packet type (3) | | | | DUP flag | QoS level | | RETAIN | Fixed Header |
| Byte 2 | Remaining Length | Remaining Length | | | | | | | | |
| Byte 3 ~ n | Topic Name | Length MSB | | | | | | | | Variable Header |
| | | Length LSB | | | | | | | | |
| | | Topic Name | | | | | | | | |
| Byte n+1 ~ m | Packet Identifier | Packet Identifier MSB | | | | | | | | |
| | | Packet Identifier LSB | | | | | | | | |
| Byte m+1 ... | Application Message | | | | | | | | | Payload |

- **DUP**标志表示该报文是第一次发送还是重复发送
- 当客户端或者服务器尝试再次发送某条**PUBLISH**报文时，必须将该标志位置1.
- 对于所有**QoS0**的消息，**DUP**必须置为0
- **QoS**表示发送消息的**QoS**级别
- **RETAIN**标志表示该条消息是否应该被服务器保存，并在有新的客户端订阅该主题的时候马上发送给该客户端
- 只有在**QoS**级别为1,2时才有**Packet Identifier**字段

| | Description | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|--------|-----------------------|------------------------------|---|---|---|----------|---|---|---|-----------------|
| Byte 1 | Message Type | MQTT Control Packet type (4) | | | | Reserved | | | | Fixed Header |
| Byte 2 | Remaining Length | Remaining Length = 2 | | | | | | | | |
| Byte 3 | Packet Identifier MSB | | | | | | | | | Variable Header |
| Byte 4 | Packet Identifier LSB | | | | | | | | | |

- Packet Identifier就是所响应的PUBLISH报文中的Packet Identifier

- SUBSCRIBE消息包含：
 - Packet ID
 - TopicName
 - QoS
 - TopicName
 - QoS
 -
- SUBACK消息包含：
 - Packet ID
 - ReturnCode 1
 - ReturnCode 2
 -



Subscribe控制报文

23

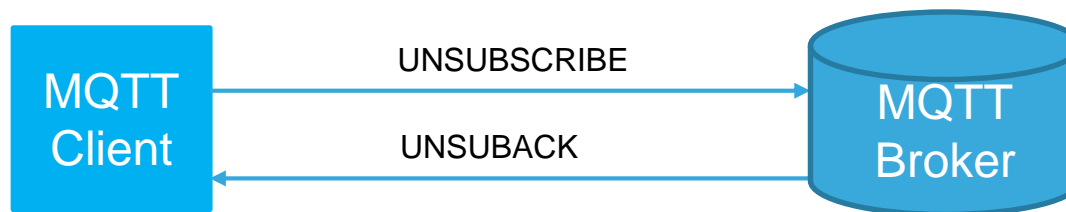
| | Description | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|------------------|-------------------|------------------------------|---|---|---|----------|-----|---|---|-----------------|
| Byte 1 | Message Type | MQTT Control Packet type (8) | | | | Reserved | | | | Fixed Header |
| Byte 2 | Remaining Length | Remaining Length | | | | | | | | |
| Byte 3 ~ 4 | Packet Identifier | Packet Identifier MSB | | | | | | | | Variable Header |
| | | Packet Identifier LSB | | | | | | | | |
| Byte 5 ~ n | Topic Filter | Length MSB | | | | | | | | Payload |
| | | Length LSB | | | | | | | | |
| | | Topic Filter | | | | | | | | |
| | Requested QoS | Reserved | | | | | QoS | | | |

- Subscribe控制报文的Payload包括一组或多组Topic Filter/Qos对
- 至少包含一组Topic Filter/Qos对
- Topic Filter支持通配符

| | Description | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|------------------|-------------------|------------------------------|---|---|---|----------|---|---|---|-----------------|
| Byte 1 | Message Type | MQTT Control Packet type (9) | | | | Reserved | | | | Fixed Header |
| Byte 2 | Remaining Length | Remaining Length | | | | | | | | |
| Byte 3 ~ 4 | Packet Identifier | Packet Identifier MSB | | | | | | | | Variable Header |
| | | Packet Identifier LSB | | | | | | | | |
| Byte 5 ~ n | Return Code | X | 0 | 0 | 0 | 0 | 0 | x | x | Payload |

- Payload是Subscribe的返回值，与Subscribe报文中的Topic Filter对应，顺序一致
- 返回值：
 - 0x00 - Success - Maximum QoS 0
 - 0x01 - Success - Maximum QoS 1
 - 0x02 - Success - Maximum QoS 2
 - 0x80 - Failure

- UNSUBSCRIBE消息包含：
 - Packet ID
 - TopicName 1
 - TopicName 2
 -
- UNSUBACK消息包含：
 - Packet ID



Unsubscribe控制报文

| | Description | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|------------------|-------------------|-------------------------------|---|---|---|----------|---|---|---|-----------------|
| Byte 1 | Message Type | MQTT Control Packet type (10) | | | | Reserved | | | | Fixed Header |
| Byte 2 | Remaining Length | Remaining Length | | | | | | | | |
| Byte 3 ~ 4 | Packet Identifier | Packet Identifier MSB | | | | | | | | Variable Header |
| | | Packet Identifier LSB | | | | | | | | |
| Byte 5 ~ n | Topic Filter | Length MSB | | | | | | | | Payload |
| | | Length LSB | | | | | | | | |
| | | Topic Filter | | | | | | | | |

- Unsubscribe控制报文的Payload包括一组或多组Topic Filter
- 至少包含一组Topic Filter
- Topic Filter支持通配符

Server ---> Client

Unsuback控制报文

27

| | Description | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|------------------|-------------------|-------------------------------|---|---|---|----------|---|---|---|-----------------|
| Byte 1 | Message Type | MQTT Control Packet type (11) | | | | Reserved | | | | Fixed Header |
| Byte 2 | Remaining Length | Remaining Length | | | | | | | | |
| Byte 3 ~ 4 | Packet Identifier | Packet Identifier MSB | | | | | | | | Variable Header |
| | | Packet Identifier LSB | | | | | | | | |

➤ Packet Identifier与Unsubscribe报文中的Packet Identifier对应

- 客户端通过CONNECT消息中的CleanSession这一位用来控制Session状态的保持期

0: 客户端和服务端断开连接后，都必须保存session的状态。

1: 客户端和服务端断开连接后，必须丢掉之前的session状态。再次连接后，重新开始新的session。

- Session状态包括：

已经发出但没有收到响应的QoS1和QoS2的消息；

客户端的订阅信息（服务器）；

还未发送给客户端的消息(服务器)；

持久化会话和消息缓存

29

1. 控制设备和实体设备1正常接入物接入，控制设备发布“QoS=1”的消息 Message001，实体设备1可以正常接收到



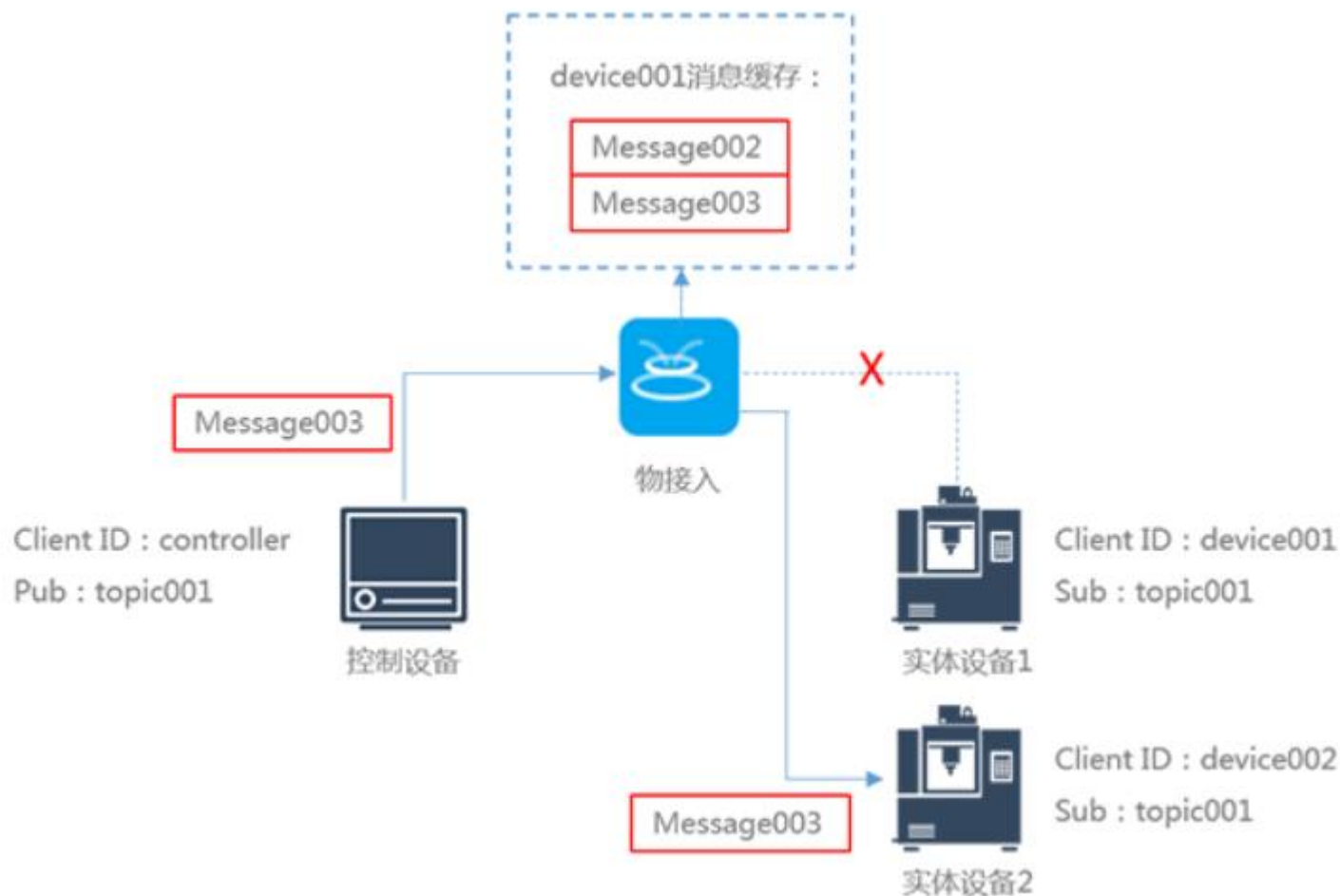
2. 如果实体设备1异常离线，此时物接入将缓存所有发送至该设备的消息。



持久化会话和消息缓存

30

3.此时如果有其它订阅了相同主题的设备连接至物接入，可以正常接收到控制设备新发送的消息，但无法接收到之前被缓存的消息



持久化会话和消息缓存

31

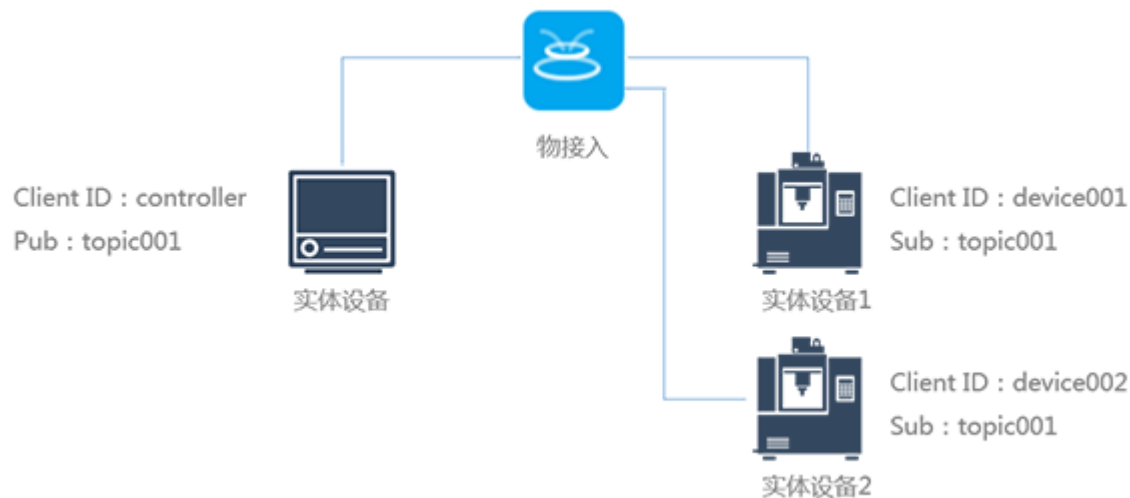
4. 实体设备1重新接入物接入，根据状态的不同，物接入有以下两种处理方式：
- 如果**Clean Session**置为**False**，此时物接入将所有缓存的消息转发至实体设备1。



持久化会话和消息缓存

32

- 如果Clean Session为True，物接入会丢弃已经缓存任何会话状态信息，为实体设备1创建一个新的会话连接。



保留(Retained)消息

33

- Retained消息就是一条加上了retained标志的普通的消息
- 每个topic只有一条Retained消息
- Retained消息可以帮首次订阅topic的客户端立刻获取最近一次的信息
- CleanSession置1并不能删除Retained消息
- 发送retained消息
 - 发布消息时将retained标志置1
- 删除retained消息
 - 发送一条payload为空的retained消息

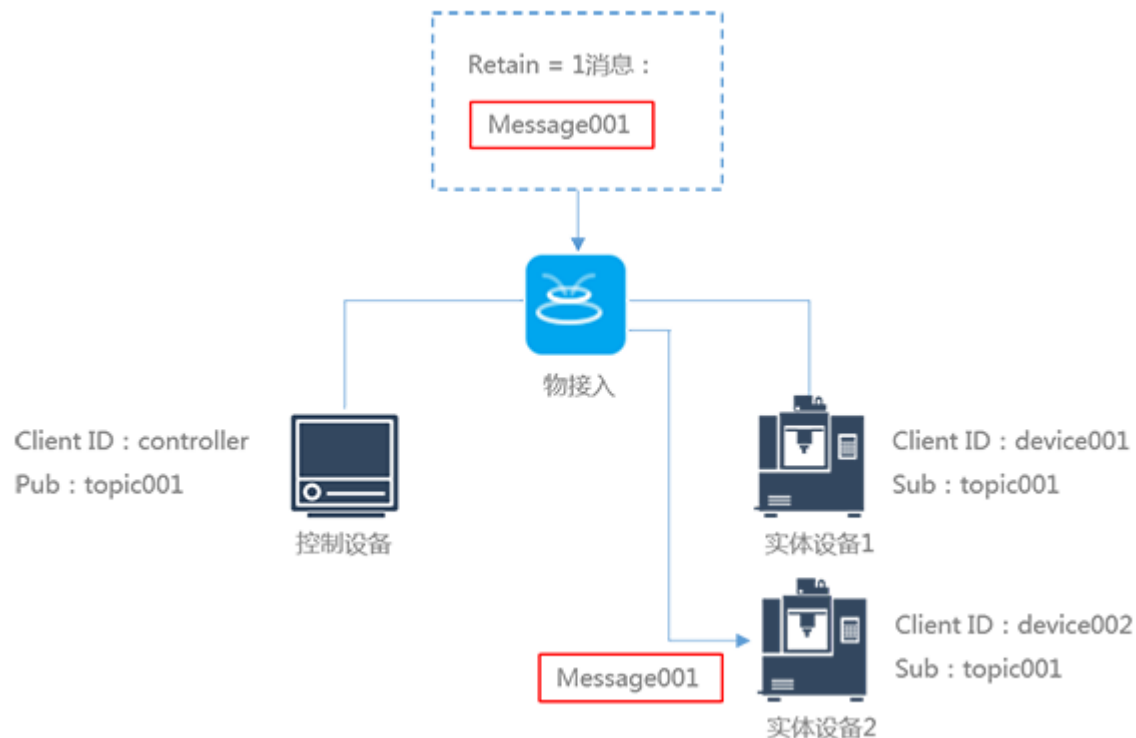
如果PUBLISH消息固定头部RETAIN标记为1，物接入会持久保存此消息，直到该消息被新的PUBLISH消息（RETAIN=1）覆盖或用户主动清除该消息。

对于“RETAIN=1”的消息，物接入不但会将该消息发送给所有当前的订阅者，同时新的接入设备也会收到该消息，如下图所示：

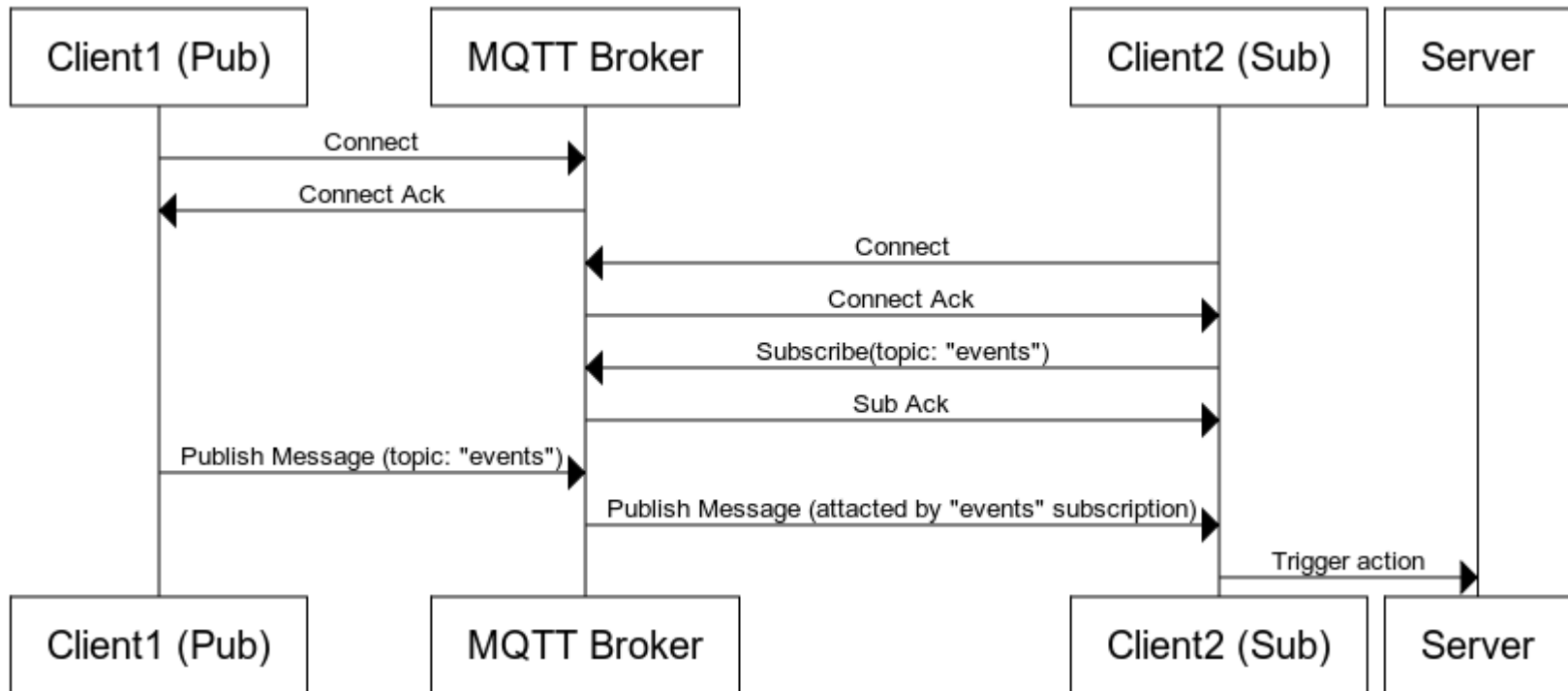
1.控制设备和实体设备1正常接入物接入，控制设备发布“Retain=1”的消息Message001，实体设备1可以正常接收到并且物接入持久保存该消息。



2.新实体设备2连接后，物接入持久保存的消息发布给新接入设备。



- 用来在发生异常断开连接时通知其他客户端
- 与Broker建立连接时，告诉Broker遗嘱消息
- Broker保存遗嘱消息，当检测到该客户端异常断开连接时通知其他的客户端
- 当客户端通过发送DISCONNECT消息来断开和服务器的连接时，Broker删除对应的遗嘱消息



- Mosquitto (<http://mosquitto.org/>)

Eclipse Mosquitto™ is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 3.1 and 3.1.1.

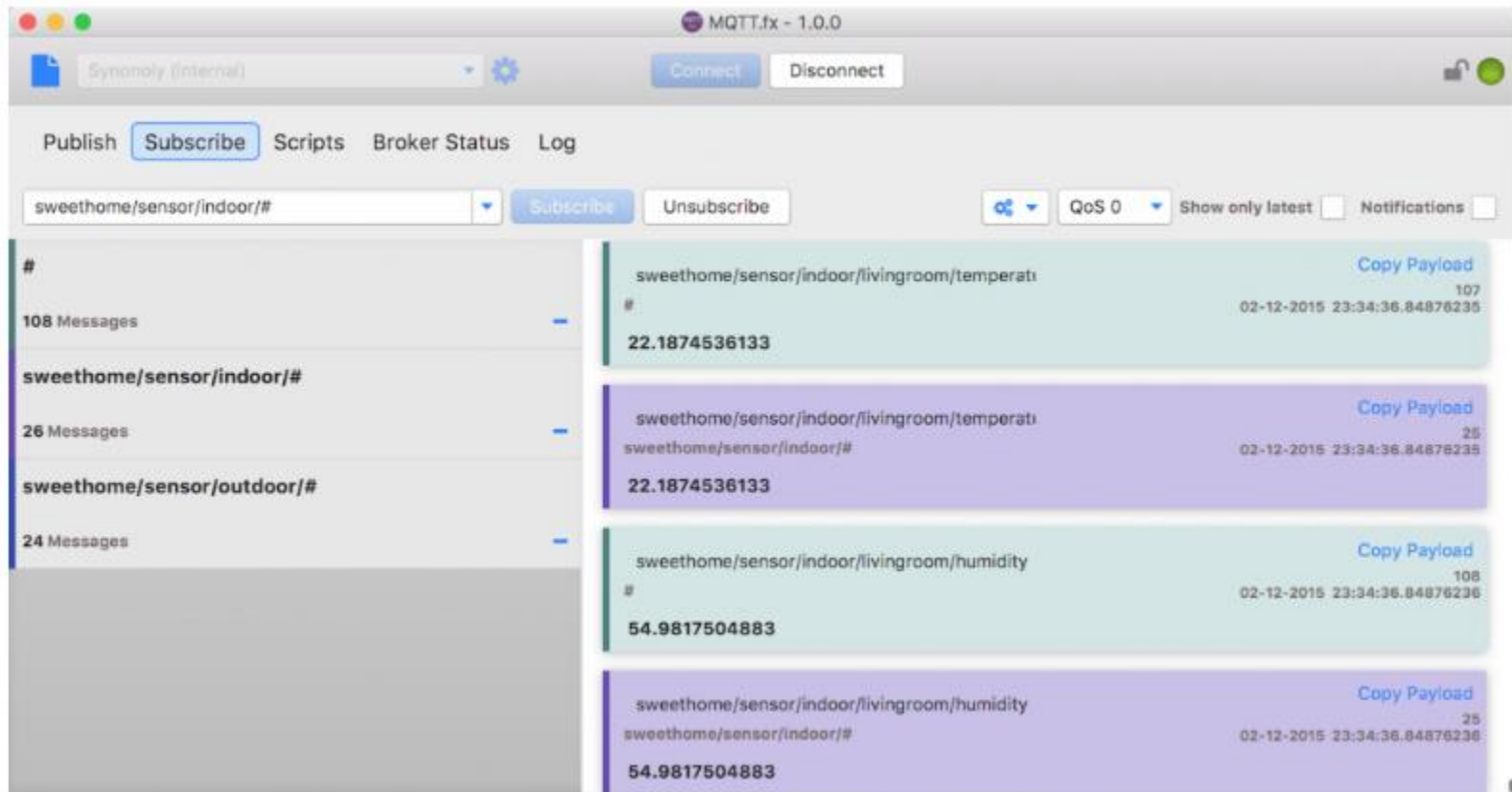
- Paho (<http://www.eclipse.org/paho/>)

The Eclipse Paho project provides open-source client implementations of MQTT and MQTT-SN messaging protocols aimed at new, existing, and emerging applications for the Internet of Things (IoT).

Eclipse Paho Downloads

Client Comparison

| Client | MQTT 3.1 | MQTT 3.1.1 | LWT | SSL / TLS | Automatic Reconnect | Offline Buffering | Message Persistence | WebSocket Support | Standard MQTT Support | Blocking API | Non-Blocking API | High Availability |
|-----------------|----------|------------|-----|-----------|---------------------|-------------------|---------------------|-------------------|-----------------------|--------------|------------------|-------------------|
| Java | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Python | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ |
| JavaScript | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✓ | ✓ |
| GoLang | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ |
| C | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ |
| .Net (C#) | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ | ✓ | ✗ |
| Android Service | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ |
| Embedded C/C++ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✗ |



- MQTT通信都是明文
- TLS为MQTT提供安全的通信通道
- 基于TLS的MQTT协议使用8883端口（实际中并没统一）

