

[illegible]

刘煜

西安中心

目录

历史

网络结构

主题

消息流程

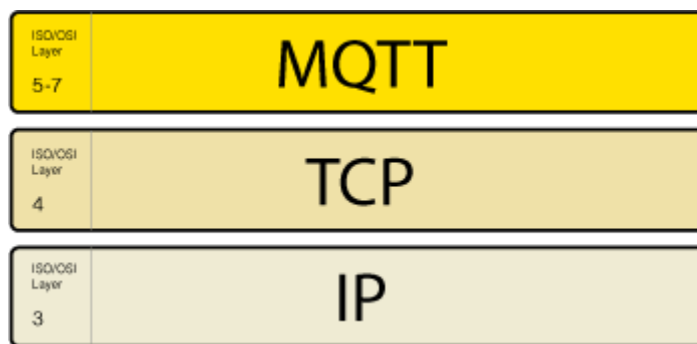
消息格式

服务器

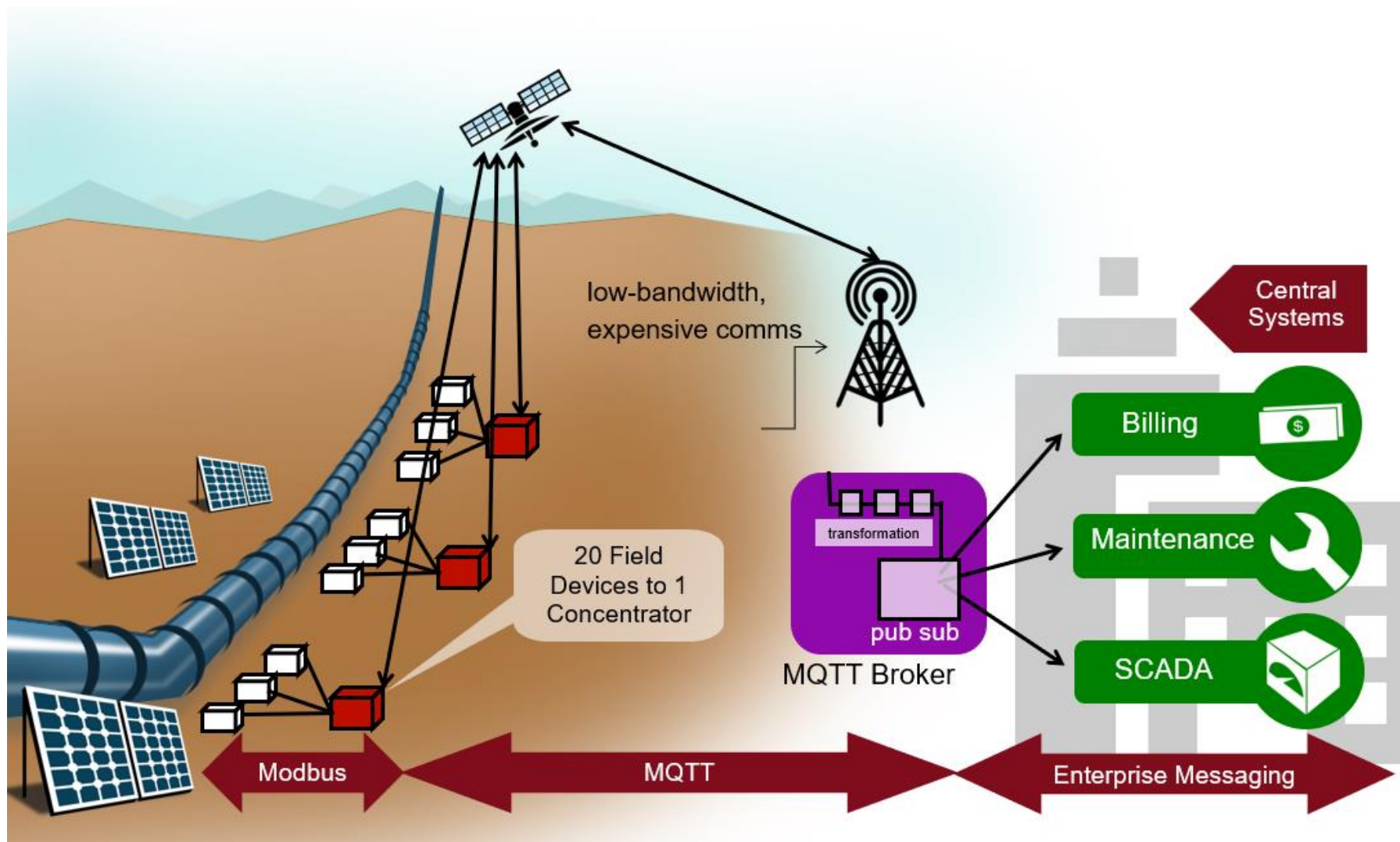
客户端编程

MQTT简介

MQTT (Message Queuing Telemetry Transport, 消息队列遥测传输协议), 是一种基于发布/订阅 (publish/subscribe) 模式的"轻量级"通讯协议, 该协议构建于TCP/IP协议上, 由IBM在1999年发布。MQTT最大优点在于, 可以以极少的代码和有限的带宽, 为连接远程设备提供实时可靠的消息服务。作为一种低开销、低带宽占用的即时通讯协议, 使其在物联网、小型设备、移动应用等方面有较广泛的应用。MQTT协议最新版本5.0 (2019/4发布), 目前主要使用的是3.1.1版本协议。

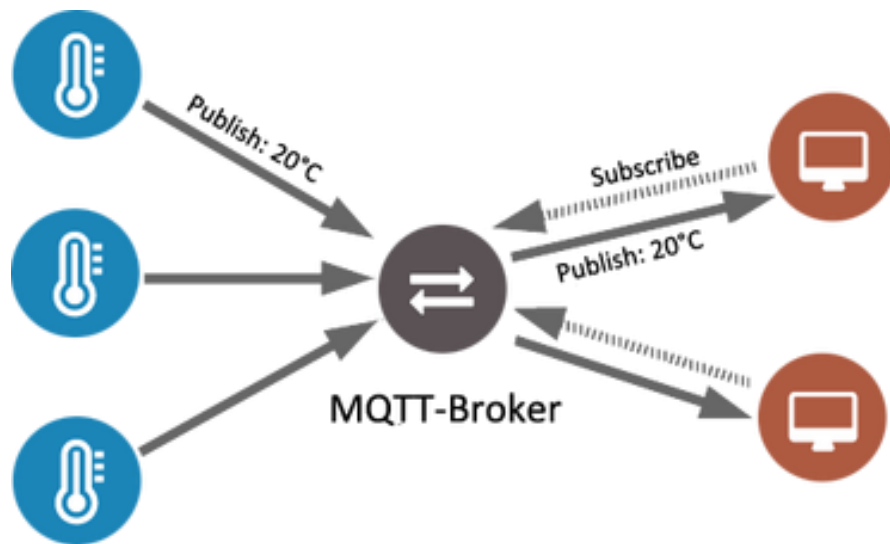


MQTT历史



MQTT网络结构

与HTTP的请求/回答这种同步模式不同，MQTT使用异步的发布/订阅（pub/sub）模式。我们将发送数据的客户端称之为发布者，将接收数据的客户端称为订阅者，发布/订阅模式将这些客户端进行解耦。这意味着发布者和订阅者互相不知道对方的存在，在其之间存在一个第三方组件，我们称之为broker（中介/代理），发布者和订阅者只与broker进行通信。MQTT Broker是MQTT服务器，发布者和订阅者是MQTT的客户端。



发布/订阅模式的优点

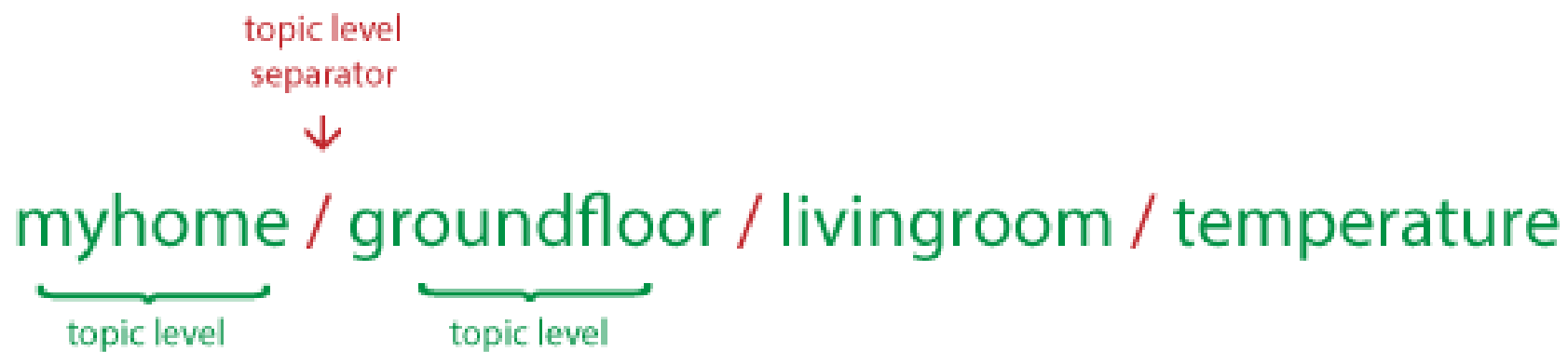
发布/订阅模式可以将发布者和订阅者在多个维度进行解耦：

- 空间解耦：发布者和订阅者不需要互相知道对方的存在，例如对方的ip地址，端口号等。由于存在Broker，客户端可以位于不同的局域网中，不受网络防火墙的限制。
- 时间解耦：发布者和订阅者不需要同时运行。
- 同步解耦：在发布和订阅消息时，双方不需要互相等待。

MQTT采用基于主题的消息过滤方式，所以每个消息都需要包含一个主题以便于broker识别，进而也决定了订阅者是否能收到这条消息。

MQTT主题

主题是一个UTF-8字符串，由一个或多个主题级别组成。每个主题级别之间由正斜杠（主题级别分隔符）分隔。



单级通配符

MQTT客户端订阅主题时可以使用通配符，通配符分为单级和多级通配符。

single-level
wildcard
↓
myhome / groundfloor / + / temperature
|
only one level

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

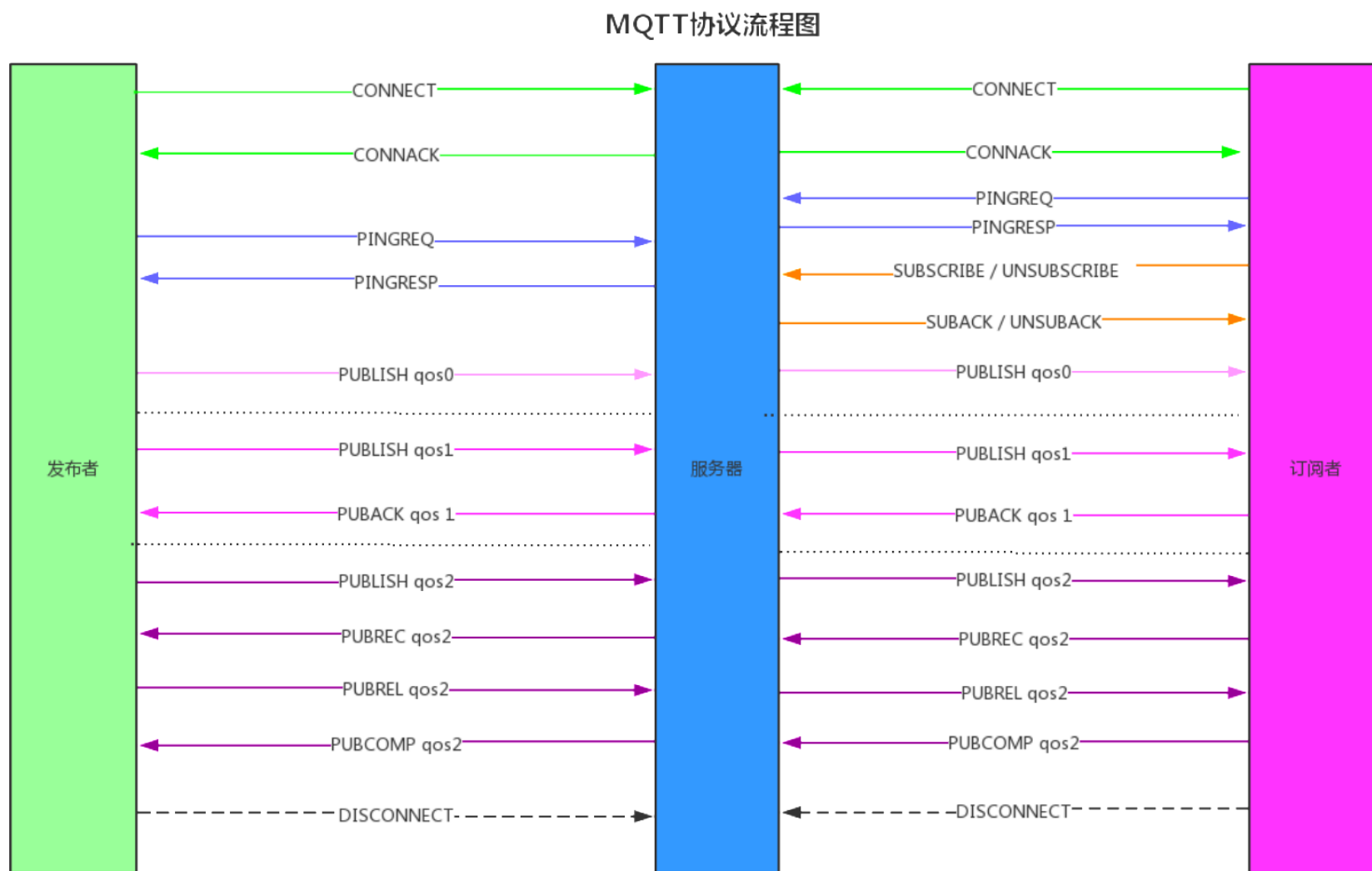
多级通配符

multi-level
wildcard
↓
myhome / groundfloor / #

only at the end
multiple topic levels

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

MQTT消息流程



| 建立连接

MQTT连接只发生在客户端和服务端之间，客户端并不直接与客户端连接。连接开始于一个客户端给服务器发送一条CONNECT消息。服务器回复一条CONNACK和状态码。一旦连接建立，服务器将保持与客户端的连接，直至客户端发送断开连接命令或者连接异常断开。

ClientId是连接到服务器的每个MQTT客户端的唯一标识符。服务器使用此标识符来识别客户端以及客户端的当前状态。

Clean session 字段表明客户端是否想与服务器建立持久的会话。一个持久的会话（cleanSession为false）意味着，当使用QoS级别为1或2时，服务器将会存储所有的客户端订阅的消息，和尚未送达的消息，此时ClientId不能为空。如果cleanSession为true时，服务器不会存储任何客户端订阅的消息，并会将之前所存的内容清空，此时ClientId 可以为空。

QoS级别

MQTT支持三种不同级别的服务质量（Quality of Service, QoS）为不同场景提供消息可靠性：

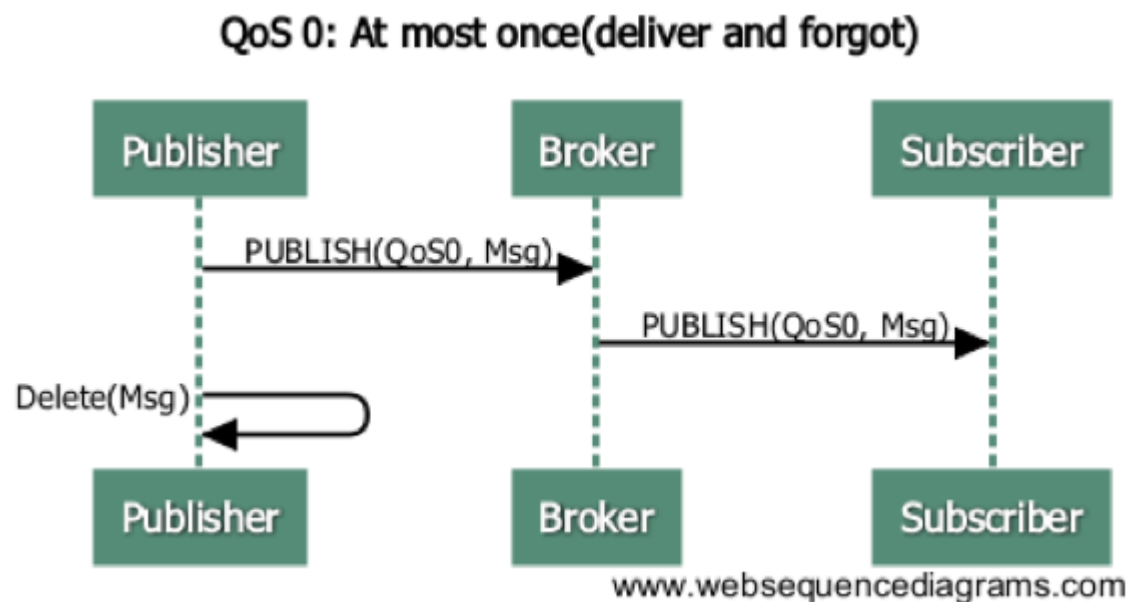
- 至多一次 (0)
- 至少一次 (1)
- 恰好一次 (2)

发布者客户端到服务器的QoS由发布者所设置的消息体中的QoS级别来确定。当服务器将消息转发给订阅者时，QoS由订阅者之前所设定的QoS级别来确定。这意味着，如果消息被一个低级别QoS的订阅者所订阅，那么消息的服务质量级别有可能会在转发时进行降级。

在同一个客户端和服务端之间的每一个数据包的标识符（用在QoS 1和QoS 2时）都是唯一的。

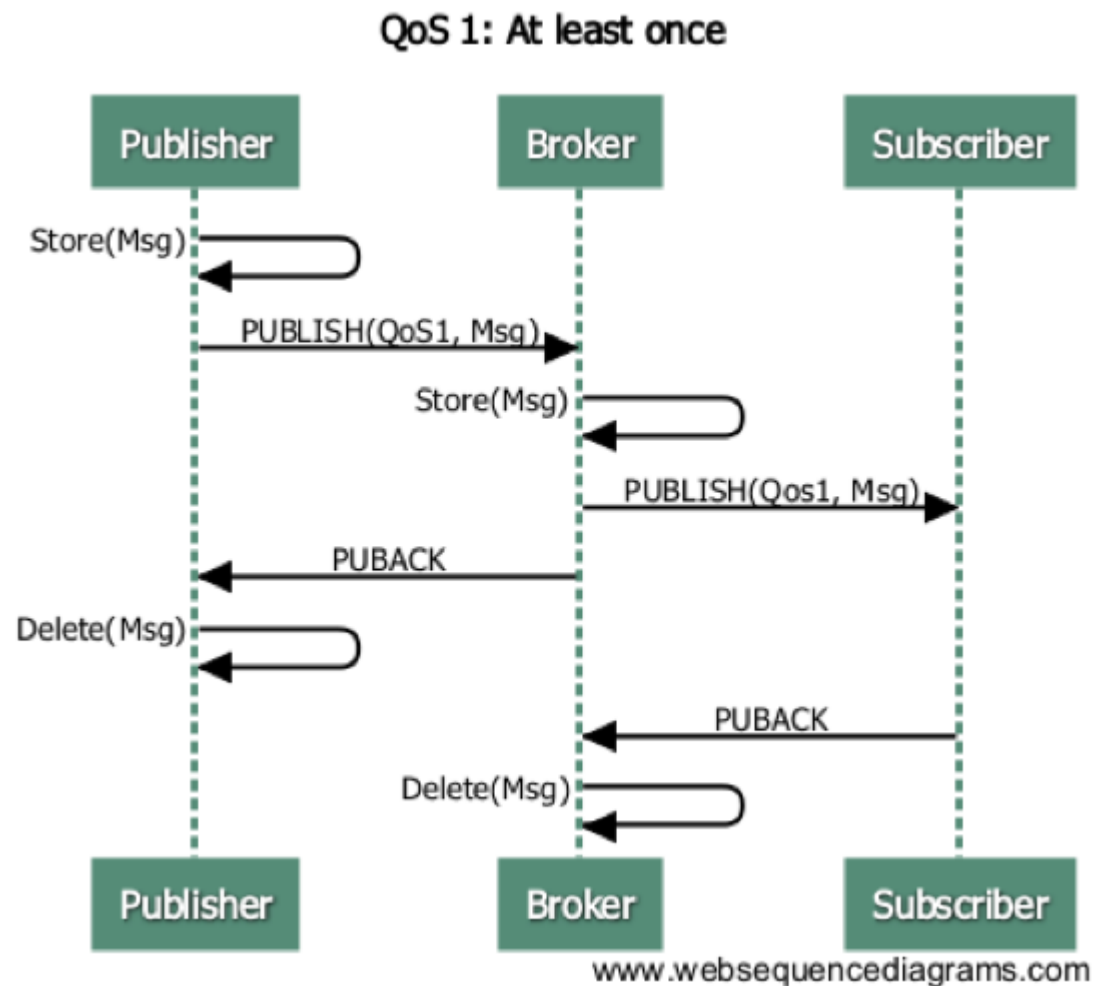
QoS 0 至多一次

级别0：尽力而为。接收者（中介或订阅者）最多收到一次消息，也可能收不到。具有最高的传输性能。接收者不会应答消息，发送者也不会保存和重发消息。



QoS 1 至少一次

级别1：至少一次。可以保证消息至少被接收者收到一次，也可能收到多次。发送者会保存消息，直至其收到接收者发送的PUBACK确认报文，如果一段时间没有收到PUBACK报文，会重发消息。

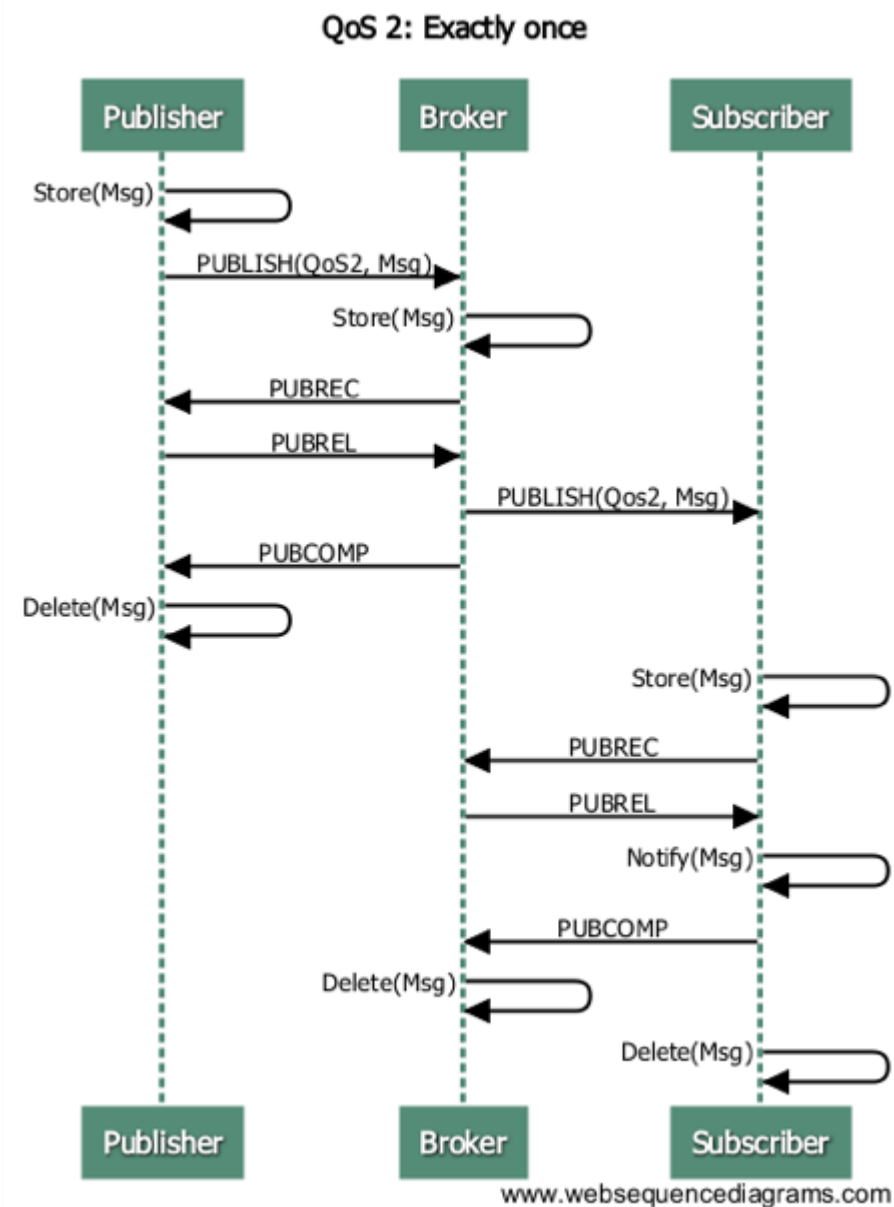


QoS 2 恰好一次

级别2：恰好一次。可以保证接收者每条消息只收到一次。

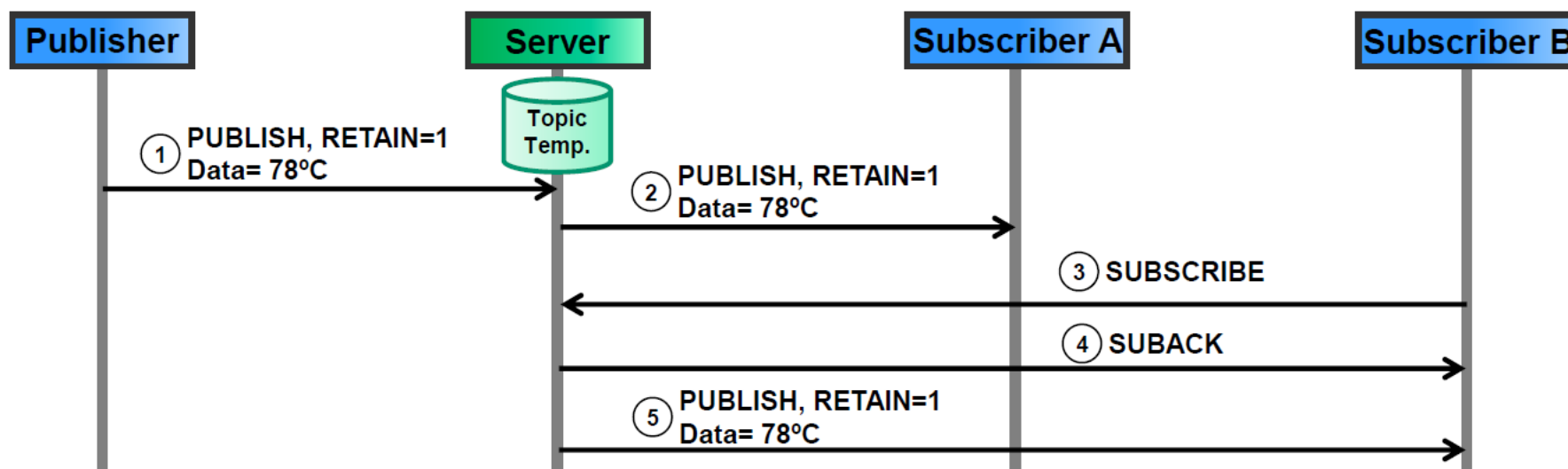
它是最安全的但也是最慢的服务级别。

- 接收者接收到一条QoS为2的PUBLISH消息后，会处理此消息并返回一条PUBREC报文。接收者会存储包标识符以作为参考，直至其发送了PUBCOMP消息。此机制避免了消息被二次处理。
- 当发送者收到PUBREC消息后，它就可以安全丢弃掉之前的发布消息，因为它已经知道接收者成功收到了消息。发布者会保存PUBREC消息并应答一个PUBREL。
- 在接收者收到PUBREL消息后，它会丢弃掉所有已经保存的状态，并应答一个PUBCOMP。同样的，当发送者收到PUBCOMP消息时也会清空之前所保存的状态。
- 无论在传输过程中何时出现丢包，发送端都负责重发上一条消息。



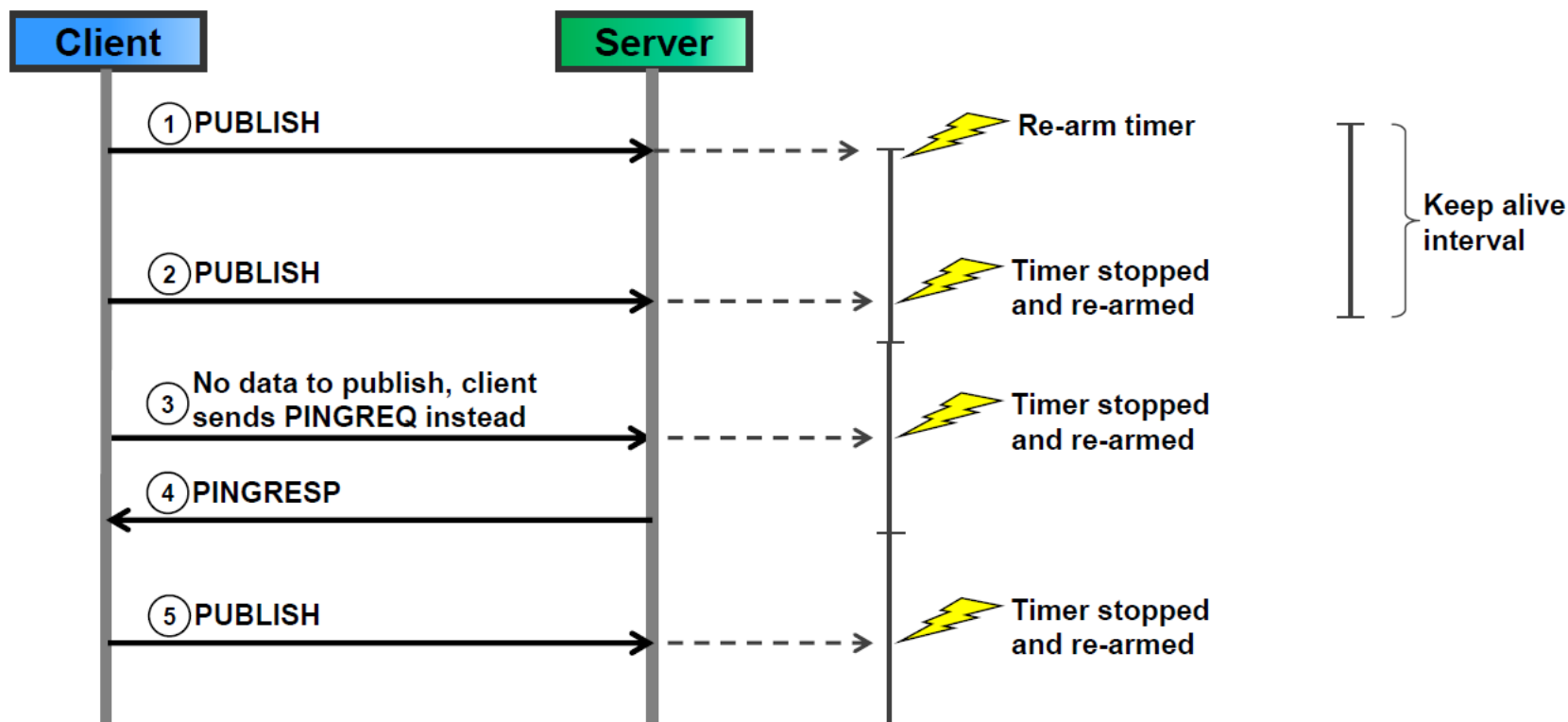
保留消息

发布者发布消息时如果设置了保留标志（Retained）服务器会针对主题依照QoS级别保留最后一条保留消息，当订阅者订阅主题时会立即收到保留消息。服务器仅为每个主题保留一条保留消息。一个主题的保留消息是最新的可知的有效数据。



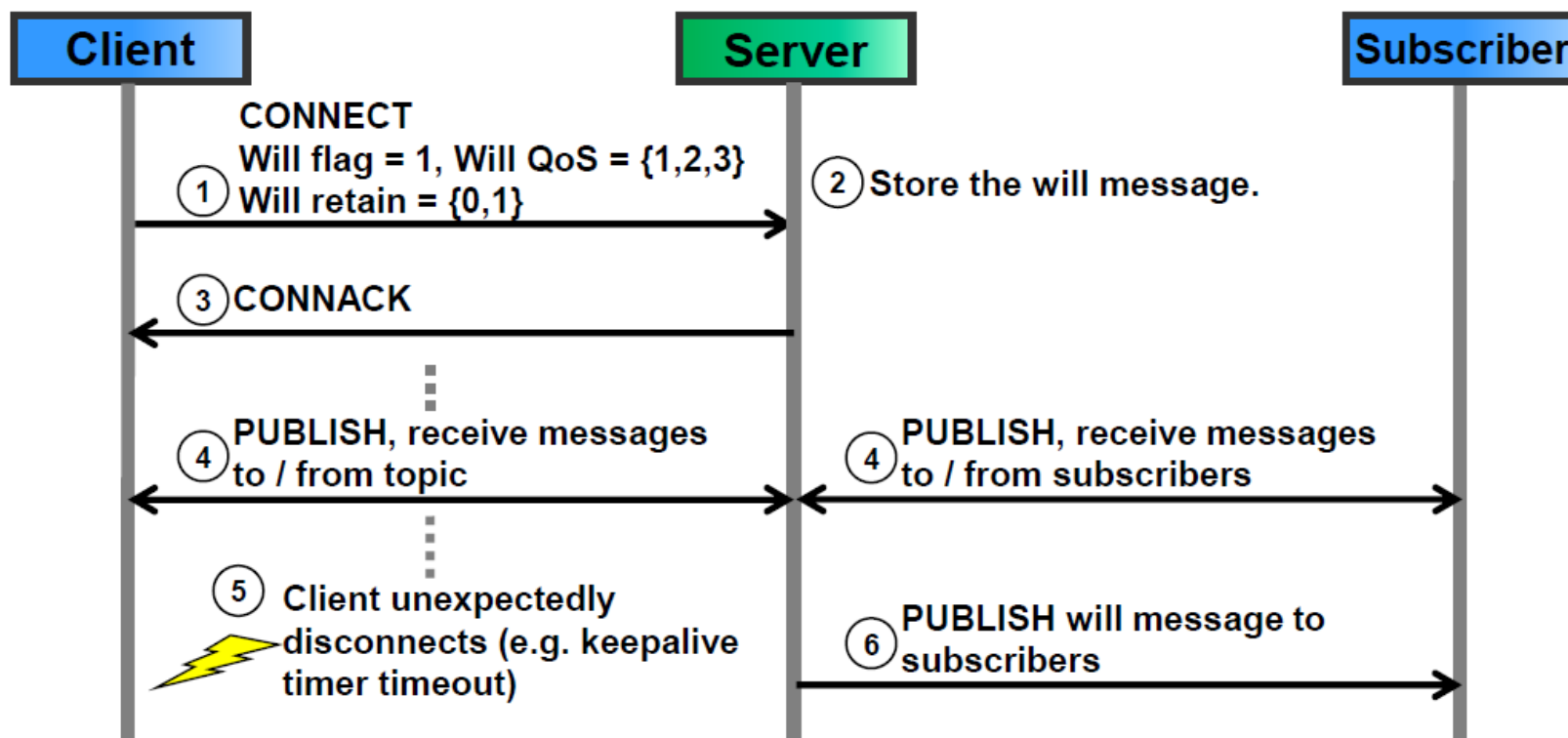
心跳机制

客户端确保发送的数据包间隔不超过心跳周期。如果没有数据包待发，那么客户端必须发送一个PINGREQ包。如果客户端没有在1.5倍心跳周期内发送PINGREQ或者其他数据，那么服务器必须与客户端断开连接。客户端决定KeepAlive时间间隔，可以是1-65535秒，为0时禁用心跳机制。



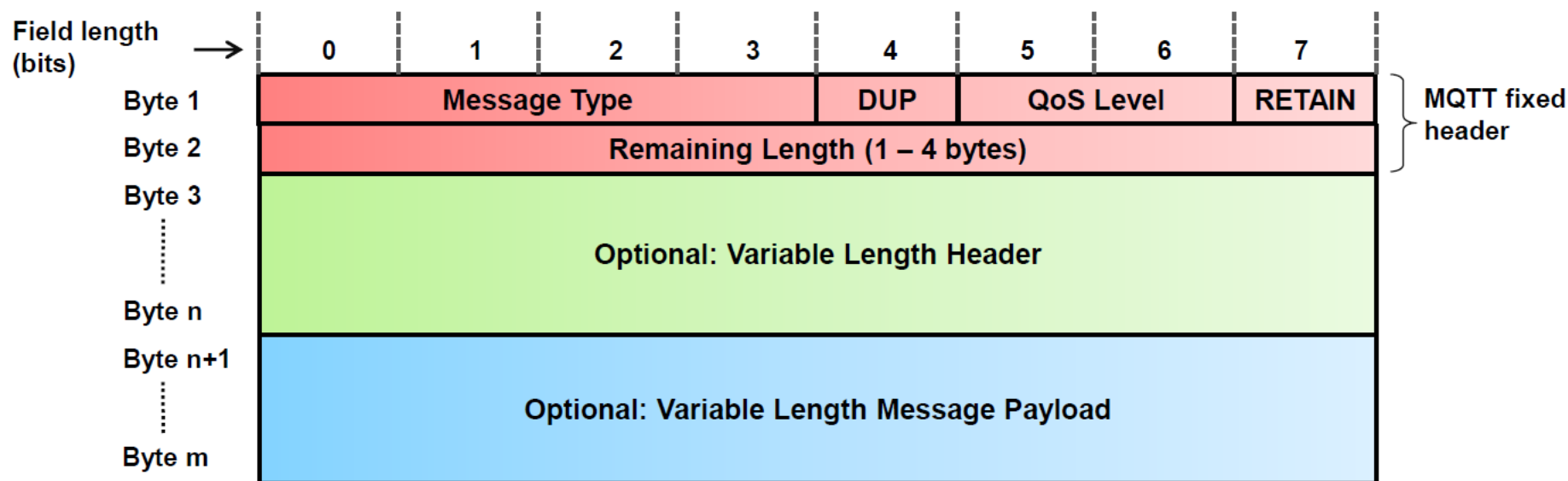
遗嘱机制

MQTT经常被用在网络不可靠的情况下。因此客户端可能会不时地意外断开连接，可能是没电了或者是其他原因。因此，我们需要知道客户端是否是正常断开连接的（是否发送了DISCONNECT消息），以便于采取适当的行动。遗嘱机制（Last Will）就是为了在发布者失联的时候通知订阅者而设计的。



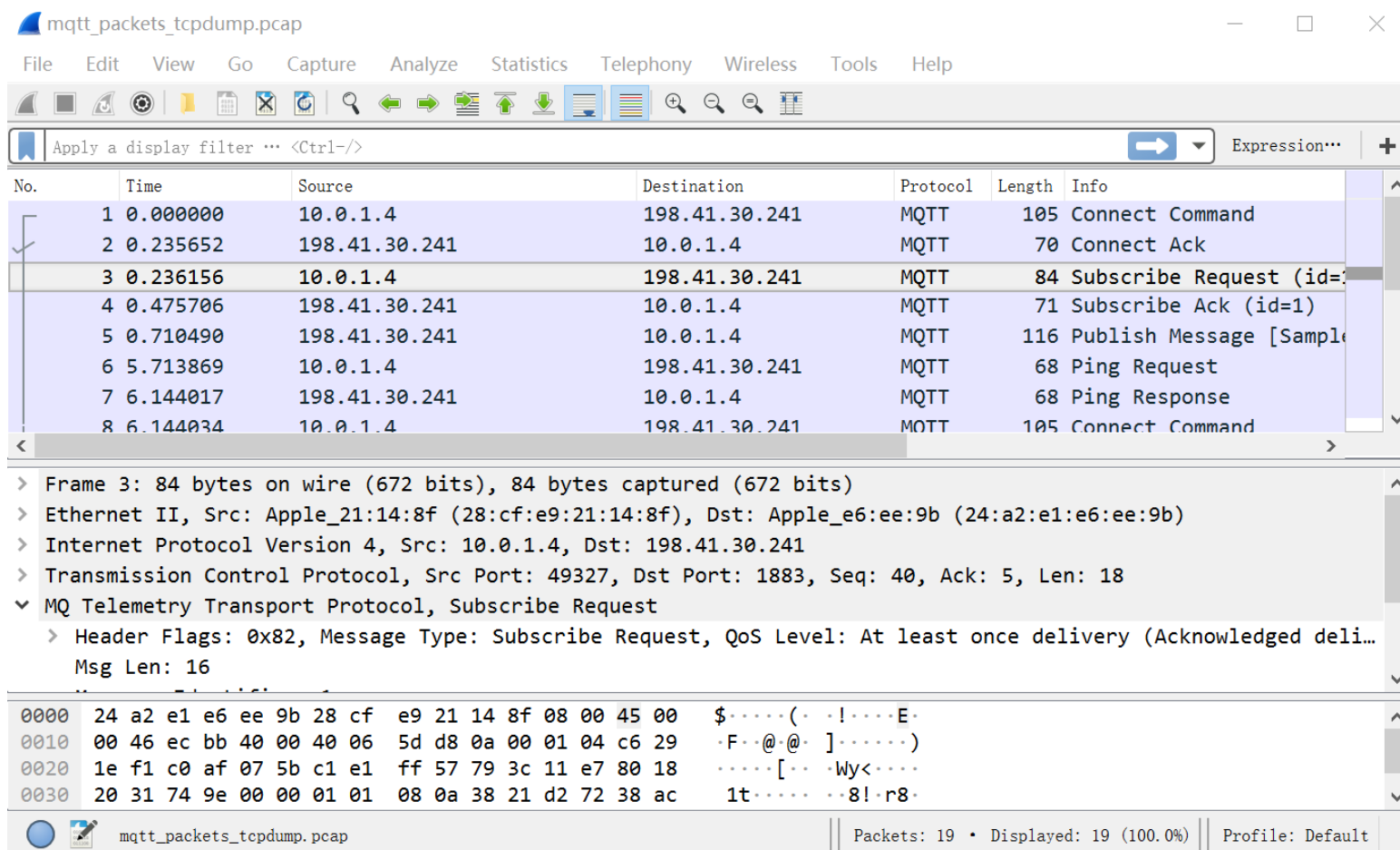
MQTT消息格式

MQTT协议控制报文的格式包含以下三个部分，以固定报头，可变报头和有效载荷，其中固定报文头是所有的控制报文都有。



消息实例

MQTT消息内容可以是二进制数据、文本、XML或JSON，完全由上层应用决定Payload的格式。



The image shows a Wireshark packet capture of MQTT traffic. The main packet list table is as follows:

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|----------|---------------|---------------|----------|--------|--------------------------|
| 1 | 0.000000 | 10.0.1.4 | 198.41.30.241 | MQTT | 105 | Connect Command |
| 2 | 0.235652 | 198.41.30.241 | 10.0.1.4 | MQTT | 70 | Connect Ack |
| 3 | 0.236156 | 10.0.1.4 | 198.41.30.241 | MQTT | 84 | Subscribe Request (id=1) |
| 4 | 0.475706 | 198.41.30.241 | 10.0.1.4 | MQTT | 71 | Subscribe Ack (id=1) |
| 5 | 0.710490 | 198.41.30.241 | 10.0.1.4 | MQTT | 116 | Publish Message [Sample] |
| 6 | 5.713869 | 10.0.1.4 | 198.41.30.241 | MQTT | 68 | Ping Request |
| 7 | 6.144017 | 198.41.30.241 | 10.0.1.4 | MQTT | 68 | Ping Response |
| 8 | 6.144034 | 10.0.1.4 | 198.41.30.241 | MQTT | 105 | Connect Command |

The details pane for the selected packet (Frame 3) shows the following structure:

- Frame 3: 84 bytes on wire (672 bits), 84 bytes captured (672 bits)
- Ethernet II, Src: Apple_21:14:8f (28:cf:e9:21:14:8f), Dst: Apple_e6:ee:9b (24:a2:e1:e6:ee:9b)
- Internet Protocol Version 4, Src: 10.0.1.4, Dst: 198.41.30.241
- Transmission Control Protocol, Src Port: 49327, Dst Port: 1883, Seq: 40, Ack: 5, Len: 18
- MQ Telemetry Transport Protocol, Subscribe Request
 - Header Flags: 0x82, Message Type: Subscribe Request, QoS Level: At least once delivery (Acknowledged deli...
 - Msg Len: 16

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0000 24 a2 e1 e6 ee 9b 28 cf e9 21 14 8f 08 00 45 00 $....(..!....E.
0010 00 46 ec bb 40 00 40 06 5d d8 0a 00 01 04 c6 29 .F..@.@. ].....)
0020 1e f1 c0 af 07 5b c1 e1 ff 57 79 3c 11 e7 80 18 .....[...Wy<....
0030 20 31 74 9e 00 00 01 01 08 0a 38 21 d2 72 38 ac 1t.....8!r8.
```

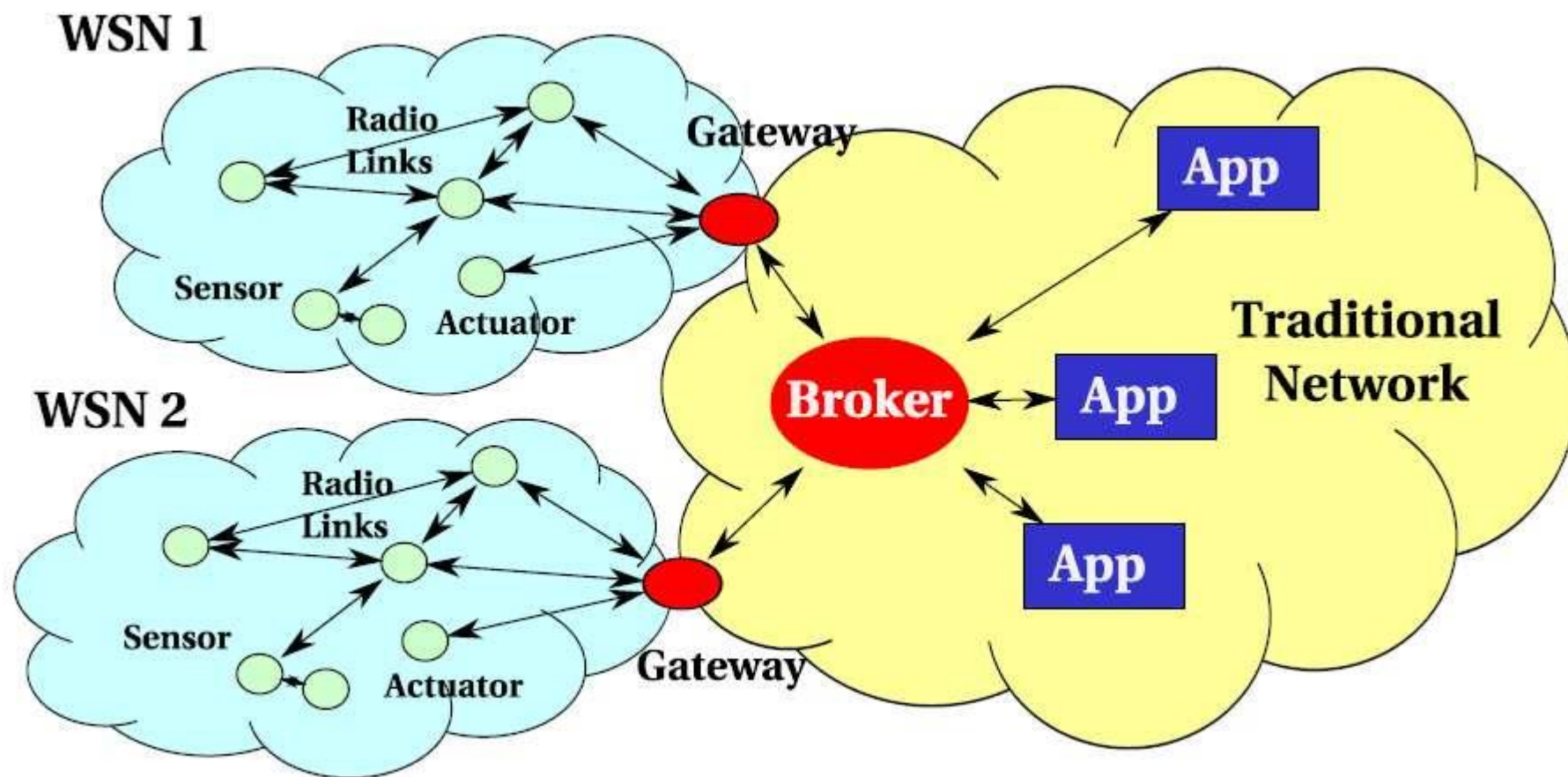
MQTT-SN

SN代表Sensor Network，针对低功耗、电池驱动、资源受限、不支持TCP协议的电子设备而定制，可以使用ZigBee，蓝牙，串口，UDP等方式进行通信。

1. MQTT-SN和MQTT大部分消息都相同，比如都有遗嘱消息，都有Connect/Subscribe/Publish消息。
2. MQTT-SN可以使用2字节的Topic ID来代替主题，需要使用REGISTER消息注册Topic ID和主题的对应关系。
3. MQTT-SN可以随时修改遗嘱的内容，甚至可以取消，而MQTT只能在建立连接时设置遗嘱的内容并且不能修改。
4. MQTT-SN网络中有网关设备，它负责把MQTT-SN消息转换成MQTT消息，和Broker进行通信。
5. 如果设备进入睡眠状态，无法接收数据，网关可以把该设备需要接收的消息缓存起来，等设备唤醒后再传送。

| | | | |
|-------------|---------------------|------|----------|
| Application | HTTP, AMQP, MQTT | CoAP | MQTT- SN |
| Transport | TCP | UDP | |
| Network | IPv4, IPv6, 6LoWPAN | | |
| Data Link | MAC, IEEE 802.15.4 | | |
| Physical | PHY | | |

MQTT-SN组网



| Mosquitto



mosquitto是一个用C语言实现的轻量级MQTT服务器和客户端，完全兼容MQTT 3.1.1。

安装：

```
apt install mosquitto mosquitto-clients
```

发布：

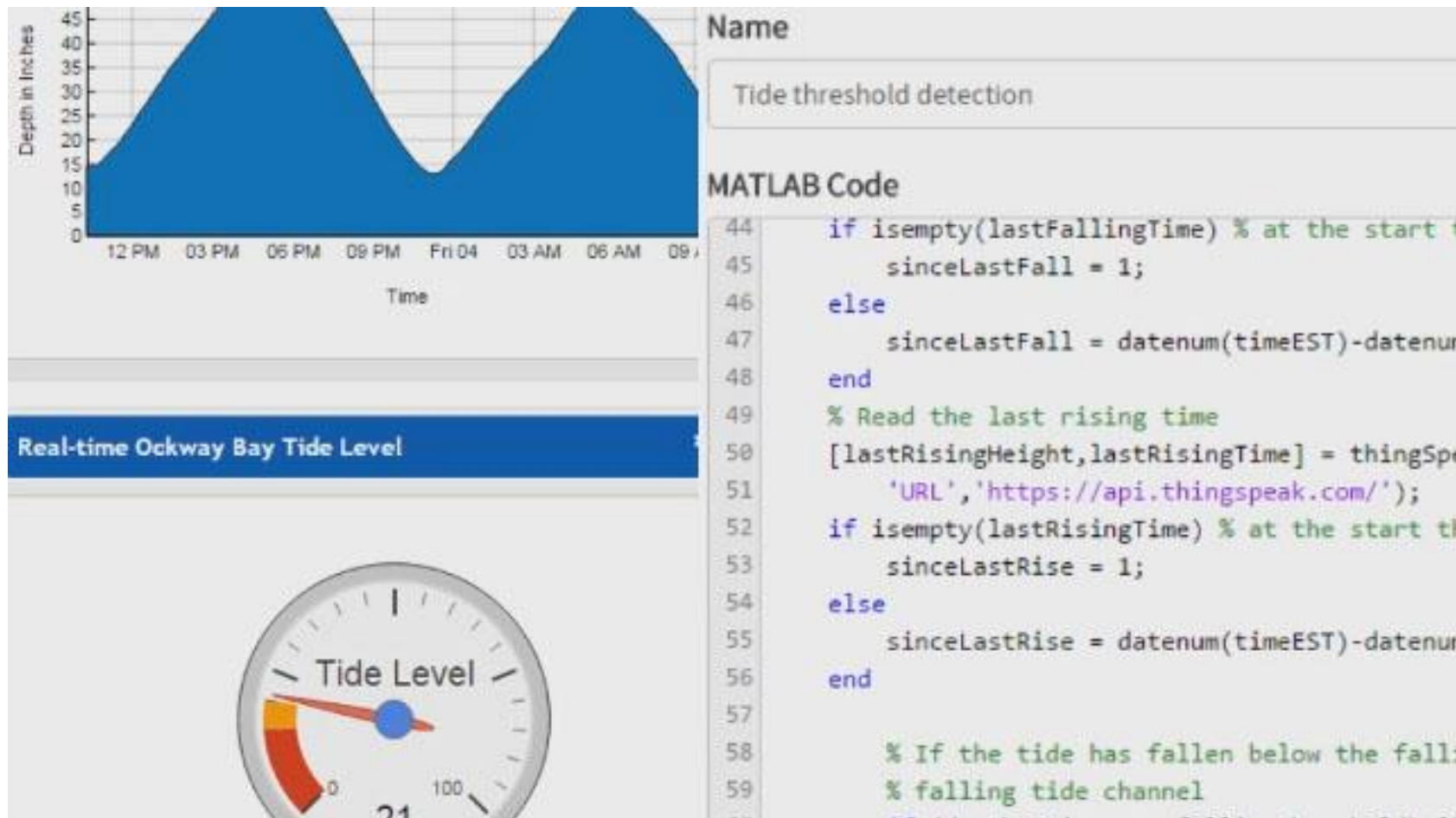
```
mosquitto_pub -h 192.168.1.1 -t sensors/temperature -m "32"
```

订阅：

```
mosquitto_sub -t sensors/temperature -q 1
```


公共MQTT服务

- test.mosquitto.org
- broker.hivemq.com
- www.cloudmqtt.com
- mqtt.flespi.io
- mqtt.thingspeak.com
- io.adafruit.com



MQTT客户端

<https://www.hivemq.com/mqtt-client-library-encyclopedia>

C: [Eclipse Paho C](#)

C++: [Eclipse Paho C++](#)

Java: [Eclipse Paho Java](#)

Javascript: [mqtt.js](#)

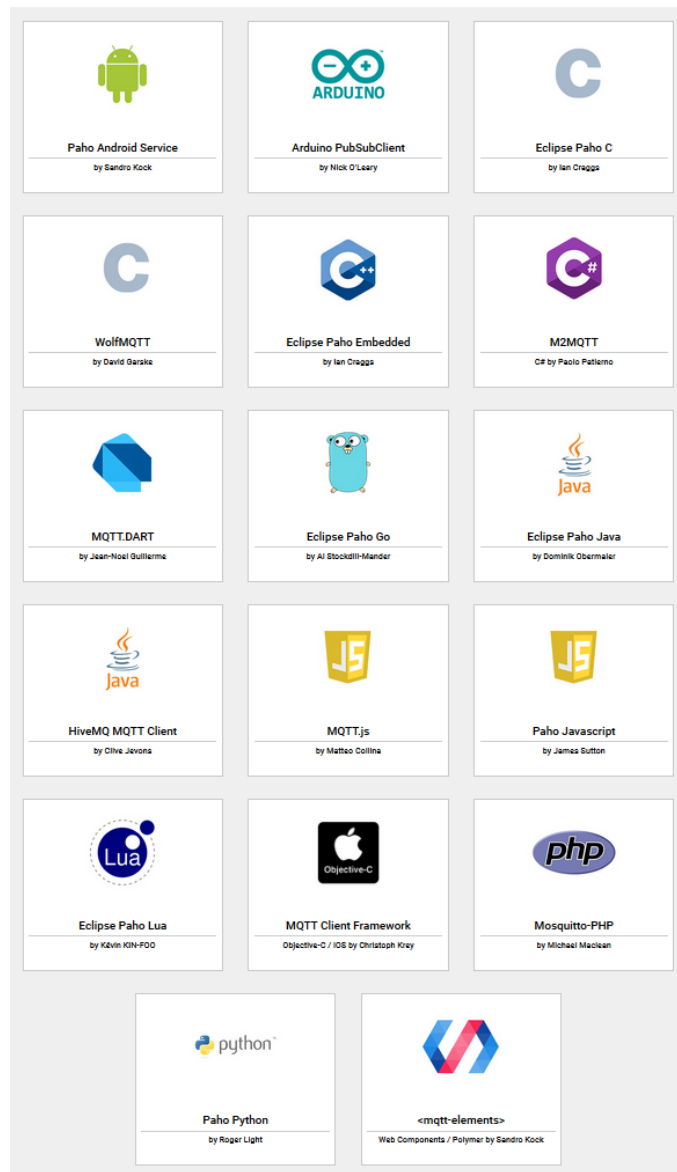
Lua: [Eclipse Paho Lua](#)

.NET: [Paho.MqttDotnet](#)

Python: [Eclipse Paho Python](#)

QT: [QtMQTT](#)

Arduino: [PubSubClient](#)





海量视频 贴身学习



超多干货 实时更新

THANKS

— 谢谢 —