# IM

**实时通信**（**I**nstant **M**essaging，简称**IM**）是一个实时[通信](https://baike.baidu.com/item/%E9%80%9A%E4%BF%A1" \t "https://baike.baidu.com/item/%E5%8D%B3%E6%97%B6%E9%80%9A%E8%AE%AF/_blank)系统，允许两人或多人使用[网络](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C/143243" \t "https://baike.baidu.com/item/%E5%8D%B3%E6%97%B6%E9%80%9A%E8%AE%AF/_blank)实时的传递[文字](https://baike.baidu.com/item/%E6%96%87%E5%AD%97/612910" \t "https://baike.baidu.com/item/%E5%8D%B3%E6%97%B6%E9%80%9A%E8%AE%AF/_blank)[消息](https://baike.baidu.com/item/%E6%B6%88%E6%81%AF/1619218" \t "https://baike.baidu.com/item/%E5%8D%B3%E6%97%B6%E9%80%9A%E8%AE%AF/_blank)、[文件](https://baike.baidu.com/item/%E6%96%87%E4%BB%B6/6270998" \t "https://baike.baidu.com/item/%E5%8D%B3%E6%97%B6%E9%80%9A%E8%AE%AF/_blank)、语音与视频交流。

## MQTT：

### 是什么

MQTT是“Message Queuing Telemetry Transport"的英文缩写，是IBM开发的一个即时通讯协议，是一个轻量级的，基于代理的”发布/订阅“模式的消息传输协议，特别适合应用于物联网通信领域，目前国内外也有很多物联网平台在部署应用。

为什么是物联网大规模使用?

MQTT就是基于TCP/IP之上建立通信的，其底层的connect机制就是TCP/IP，如果我们对物联网这种应用场景，采用TCP/IP进行自下而上的开发，需要做大量其它的工作，比如client客户端，服务器端程序，以及对网络延迟、即时性以及用户是否在线的处理，而这些MQTT都已经做了大量的工作和优化，这时采用MQTT，它的优势就显现出来了。

MQTT属于哪一层协议?

MQTT协议运行于TCP之上，属于应用层协议，因此只要是支持TCP/IP协议栈的地方，都可以使用MQTT。

### MQTT消息格式

每条MQTT命令消息的消息头都包含一个固定的报头,有些消息会携带一个可变报文头和一个负荷,消息格式如下:

**固定报文头**|**可变报文头**|**负荷**

固定报文头(Fixed Header):

MQTT固定报文头最少有两个字节

1.第一字节包含消息类型以及Qos级别等标志位

2.第二字节是剩余长度字段,该长度是后面的可变报文头加消息负载的总长度,该字段最多允许四个字节*(告诉你还有多少数据)*

由于MQTT协议最多只允许使用四个字节表示剩余长度，并且最后一字节最大值只能是0x7F(127)不能是0xFF(255)，所以能发送的最大消息长度是256MB，而不是512MB。

可变报文头(Variable Header):

可变报文头主要包含协议名,协议版本,连接标志,心跳间隔时间,连接返回码,主题名等

有效负荷(Payload):

Payload直译为负荷,可以理解为消息主体

当MQTT发送的消息类型是Connect(连接),Publish(发布),Suscribe(订阅),suback(订阅确认),Unsubscribe(取消订阅) 时,都会带有消息主体

### 主要特性

#### 消息类型(Message Type)

MQTT的消息类型(Message Type)

固定报文头中的第一个字节包含连接标识(Connect Flags),连接标志用来区分MQTT的消息类型,MQTT协议拥有14种不同的消息类型,可简单分为连接及终止、发布和订阅、QoS 2消息的机制以及各种确认ACK。



#### 消息质量(QoS)

MQTT提供了三种QoS机制，第一种是最多一次传输（At most once delivery），这种方式可能会有数据丢包，**适用于允许个别数据丢失的场景**；

第二种是最少一次传输（At least once delivery），这种方式**可能会有数据包的重复**；

第三种是精准一次传输（Exactly once delivery），这种方式**只能传输一次并且保证送达**，适用于金融支付类不允许丢包和重复的应用场景。

比如目前流行的共享单车智能锁，智能锁可以定时使用QoS level 0质量消息请求服务器，发送单车的当前位置，如果服务器没收到也没关系，反正过一段时间又会再发送一次。之后用户可以通过App查询周围单车位置，找到单车后需要进行解锁，这时候可以使用QoS level 1质量消息，手机App不断的发送解锁消息给单车锁，确保有一次消息能达到以解锁单车。最后用户用完单车后，需要提交付款表单，可以使用QoS level 2质量消息，这样确保只传递一次数据，否则用户就会多付钱了。

#### 遗愿标志(Will Flag)

在可变报文头的连接标志位字段里有三个Will标志位: Will Flag , Will QoS 和Will Retain Flag,这些Will字段用于监控客户端与服务端之间的连接状况,如果设置了Will Flag,就必须设置Will Qos和Will Retain标志位,消息主体中也必须有Will Topic与Will Message字段

遗愿消息是什么?

服务器与客户端通信时，当遇到异常或客户端心跳超时的情况，MQTT服务器会替客户端发布一个Will消息。当然如果服务器收到来自客户端的DISCONNECT(断开连接)消息，则不会触发Will消息的发送。

因此，Will字段可以应用于设备掉线后需要通知用户的场景。

#### 连接保活心跳机制(Keep Alive Timer)

MQTT客户端可以设置一个心跳间隔时间(Keep Alive Timer),表示在每个心跳间隔时间内发送一条消息,如果在这个时间周期内,没有业务数据相关的消息,客户端会发送一个Pingrep消息,相对应的,服务器会返回一个Pingresp消息进行确认,如果服务器在一个半(1.5)心跳间隔时间周期内没有收到来自客户端的消息,就会选择主动断开与客户端的连接,心跳间隔时间最大值可以设置为18个小时,0则代表客户端不断开

### MQTT其他特点

#### 异步发布/订阅实现

发布/订阅模式解耦了发布消息的客户(生产者)与订阅消息的客户(消费者)之间的关系,这意味着发布者与订阅者之间并不需要直接建立联系.

这个模式有以下好处:

生产者和消费者只需要知道同一个消息代理即可

生产者和消费者并不需要直接交互

生产者和消费者并不需要同时在线

由于采用了发布/订阅实现,MQTT可以双向通信.也就是说MQTT支持服务端反向控制设备,设备可以订阅某个主题,然后发布者对该主题发布消息,设备收到消息后即可进行一系列操作

#### 二进制格式实现

Mqtt是基于二进制实现而不是字符串,比如HTTP和XMPP都是基于字符串实现,由于HTTP和XMPP拥有冗长的协议头部,而MQTT固定报文头仅有两字节,所以相比其他协议,发送一条消息最省流量

#### MQTT的安全

由于MQTT运行于TCP层之上并以明文方式进行传输,就相等于HTTP的明文传输,可能会产生风险,比如设备被盗用,数据被篡改,报文注入,通信重定向等

作为传输协议,MQTT仅关注消息传输,提供合适的安全功能是开发者的责任,安全功能可以从三个层次来考虑---应用层|传输层| 网络层

HTTP | TCP/UDP | IP

应用层: 在应用层之上,MQTT提供了客户标识以及用户名和密码,可以在应用层验证设备**(比如我们的IM就使用userId和设备号去Broker注册获取唯一标识token用来连接MQTT)**

传输层: 类似于HTTPS,MQTT基于TCP连接,也可以加上一层TLS,传输层使用TLS加密是确保安全的一个好手段,可以防止中间人攻击,客户端证书不但可以作为设备的身份凭证,还可以用来验证设备

网络层: 如果有条件的化,可以通过拉取专线或者VPN来连接设备与MQTT代理,以提高网络传输的安全性

### MQTT与其他协议

#### MQTT与TCP Socket

由于TCP协议有粘包和分包问题，所以传输数据时需要自定义协议，如果传输的数据报超过MSS（最大报文段长度 ipv4 536字节 ipv4 1220字节），一定要给协议定义一个消息长度字段，确保接收端能通过缓冲完整收取消息。一个简单的协议定义：消息头部+消息长度+消息正文。

当然，使用MQTT协议则不需要考虑这个问题，这些MQTT都已经处理好了，MQTT最长可以一次性发送256MB数据，不用考虑粘包分包的问题。

总之，TCP和MQTT本身并不矛盾，只不过基于Socket开发需要处理更多的事情，而且大多数嵌入式开发模块本身也只会提供Socket接口供厂家自定义协议。

#### MQTT与HTTP

HTTP最初的目的是提供一种发布和接受HTML页面的方法,主要用于Web,HTTP是典型的C/S通讯模式:请求从客户端发出,服务端只能被动接受,一条连接只能发送一次请求,获取响应后就断开连接,该协议最早是为了适用Web浏览器的上网浏览场景而设计的,目前在PC,手机,Pad等终端上都应用广泛,由于这样的通信特点,HTTP技术在物联网设备中很难实现设备的反向控制

如果要实现设备的反向控制,可能就要用到前面提到的Coment技术,由于需要不断的请求服务器,会导致通信开销非常大,加上HTTP冗长的报文头,在节省流量上实在没有优势

#### MQTT与XMPP

最有可能与MQTT竞争的是XMPP协议。XMPP（可扩展通讯与表示协议）是一项用于实时通讯的开放技术，它使用可扩展标记语言（XML）作为交换信息的基本格式。其优点是协议成熟、强大、可扩展性强。目前主要应用于许多聊天系统中，在消息推送领域，MQTT和XMPP互相竞争。下面列举MQTT与XMPP各自的特性：

1. XMPP协议基于繁重的XML，报文体积大且交互繁琐；而MQTT协议固定报头只有两个字节，报文体积小、编解码容易；
2. XMPP基于JID的点对点消息传输；MQTT协议基于主题(Topic)发布\订阅模式，消息路由更为灵活；
3. XMPP协议采用XML承载报文，二进制必须进行Base64编码或其他方式处理；MQTT协议未定义报文内容格式，可以承载JSON、二进制等不同类型报文，开发者可以针对性的定义报文格式；
4. MQTT协议支持消息收发确认和QoS保证，有更好的消息可靠性保证；而XMPP主协议并未定义类似机制；
5. 在嵌入式设备开发中大多使用的是C语言开发，C语言解析XML是非常困难的。MQTT基于二进制实现且未定义报文内容格式，可以很好的兼顾嵌入式C语言开发者；而XMPP基于XML，开发者需要配合协议格式，不能灵活开发。

综上所述，在嵌入式设备中，由于需要一个灵巧简洁，对设备开发者和服务端开发者都友好的协议，MQTT比XMPP更具有优势。

### 5.MQTT总结

1. MQTT基于异步发布/订阅的实现解耦了消息发布者和订阅者
2. 基于二进制的实现节省了存储空间及流量
3. 同时MQTT拥有更好的消息处理机制，可以替代TCP Socket一部分应用场景。
4. 相对于HTTP和XMPP，MQTT可以选择用户数据格式，解析复杂度低，同时MQTT也可用于手机推送等领域。手机作为与人连接的入口，正好建立了人与物的连接，可谓一箭双雕。