MQTT协议

同济大学软件学院

MQTT协议

MQTT是什么？

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发布，目前最新版本为v5.01。MQTT最大的优点在于可以以极少的代码和有限的带宽，为远程设备提供实时可靠的消息服务。做为一种低开销、低带宽占用的即时通讯协议，MQTT在物联网、小型设备、移动应用等方面有广泛的应用。当然，在物联网开发中，MQTT不是唯一的选择，与MQTT互相竞争的协议有XMPP和CoAP协议等。

MQTT协议

MQTT层次

众所周知，TCP/IP参考模型可以分为四层：应用层、传输层、网络层、链路层。TCP和UDP位于传输层，应用层常见的协议有HTTP、FTP、SSH等。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

MQTT协议

MQTT层次

固定报文头（Fixed Header）

MQTT固定报文头最少有两个字节，第一字节包含消息类型（Message Type）和QoS级别等标志位。第二字节开始是剩余长度字段，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。剩余长度字段单个字节最大值为二进制0b0111 1111，16进制0x7F。也就是说，单个字节可以描述的最大长度是127字节。为什么不是256字节呢？因为MQTT协议规定，单个字节第八位（最高位）若为1，则表示后续还有字节存在，第八位起“延续位”的作用。

MQTT协议

MQTT层次

可变报文头（Variable Header）主要包含协议名、协议版本、连接标志（Connect Flags）、心跳间隔时间（Keep Alive timer）、连接返回码（Connect Return Code）、主题名（Topic Name）等，后面会针对主要部分进行讲解。

有效负荷（Payload）可能让人摸不着头脑，实际上可以理解为消息主体（body）。当MQTT发送的消息类型是CONNECT（连接）、PUBLISH（发布）、SUBSCRIBE（订阅）、SUBACK（订阅确认）、UNSUBSCRIBE（取消订阅）时，则会带有负荷。

MQTT的主要特性

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志（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（如下表），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。至于每一个消息类型会携带什么内容。

MQTT的主要特性

类型名称	类型值	流动方向	报文说明
Reserved	0	禁止	保留
CONNECT	1	客户端到服务器	发起连接
CONNACK	2	服务端到客户端	连接确认
PUBLISH	3	两个方向都允许	发布消息
PUBACK	4	两个方向都允许	Qos1消息确认
PUBREC	5	两个方向都允许	QoS2消息回执（保证交付第一步）
PUBREL	6	两个方向都允许	QoS2消息释放（保证交付第二步）
PUBCOMP	7	两个方向都允许	QoS2消息完成（保证交付第三步）
SUBSCRIBE	8	客户端到服务端	订阅请求
SUBACK	9	服务端到客户端	订阅确认
UNSUBSCRIBE	10	客户端到服务端	取消订阅
UNSUBACK	11	服务端到客户端	取消订阅确认
PINGREQ	12	客户端到服务端	心跳请求
PINGRESP	13	服务端到客户端	心跳响应
DISCONNECT	14	客户端到服务端	断开连接
Reserved	15	禁止	保留

MQTT的主要特性

消息质量（QoS）：MQTT消息质量有三个等级，QoS 0，QoS 1和 QoS 2。

QoS 0：最多分发一次。消息的传递完全依赖底层TCP/IP网络，协议里没有定义应答和重试，消息要么只会到达服务端一次，要么根本没有到达。

QoS 1：至少分发一次。服务器的消息接收由PUBACK消息进行确认，如果通信链路或发送设备异常，或者指定时间内没有收到确认消息，发送端会重发这条在消息头中设置了DUP位的消息。

QoS 2：只分发一次。这是最高级别的消息传递，消息丢失和重复都是不可接受的，使用这个服务质量等级会有额外的开销。

MQTT的主要特性

通过下面的例子可以更深刻的理解上面三个传输质量等级。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。

MQTT协议

遗愿标志（Will Flag）

在可变报文头的连接标志位字段（Connect Flags）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器之间的连接状况。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和Will Message字段。

那遗愿消息是怎么回事呢？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。

MQTT协议

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，客户端会发一个PINGREQ消息，相应的，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，就会断开与客户端的连接。心跳间隔时间最大值大约可以设置为18个小时，0值意味着客户端不断开。

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（发布者）与订阅消息的客户（订阅者）之间的关系，这意味着发布者和订阅者之间并不需要直接建立联系。

这个模式有以下好处：

发布者与订阅者只需要知道同一个消息代理即可；

发布者和订阅者不需要直接交互；

发布者和订阅者不需要同时在线。

MQTT协议

MQTT其他特点

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协议

有关MQTT的云平台

目前，百度、阿里、腾讯的云平台都逐渐有了物联网开发套件：腾讯QQ物联平台内测中，阿里云物联网套件公测中，两者都需要进行申请试用，而百度云物联网套件已经支持MQTT并且可以免费试用一段时间。除了BAT三大家，一些其他支持MQTT的物联网云平台。OneNET云平台：OneNET是由中国移动打造的PaaS物联网开放平台。平台能够帮助开发者轻松实现设备接入与设备连接，快速完成产品开发部署，为智能硬件、智能家居产品提供完善的物联网解决方案。OneNET云平台已经于2014年10月正式上线。云巴：云巴（Cloud Bus）是一个跨平台的双向实时通信系统，为物联网、App和Web提供实时通信服务。云巴基于MQTT，支持Socket.IO协议，支持RESTful API。

MQTT协议

MQTT与其他协议

目前各大平台都开始支持MQTT协议，MQTT相比其他协议有什么优势呢？物联网设备能不能用其他的协议呢？

MQTT与TCP Socket

虽然MQTT运行于TCP层之上，看起来这两者之间根本没有比较性，但笔者觉得还是有必要叙述一番，因为大多数从事硬件或嵌入式开发的工程师，都是直接在TCP层上通信的。从事嵌入式开发工作的人都应该知道LwIP，LwIP是一套用于嵌入式系统的开放源代码TCP/IP协议栈，LwIP在保证嵌入式产品拥有完整的TCP/IP功能的同时，又能保证协议栈对处理器资源的有限消耗，其运行一般仅需要几十KB的RAM和40KB左右的ROM。也就是说，只要是嵌入式产品使用了LwIP，就支持TCP/IP协议栈，进而可以使用MQTT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），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协议

MQTT与HTTP

HTTP最初的目的是提供一种发布和接收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，服务端只能被动接收，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，不过非要实现也不是不行，下面看一下Web端的例子。

目前，在微博等SNS网站上有海量用户公开发布的内容，当发布者发布消息，数据传到服务器更新时，就需要给关注者尽可能的实时更新内容。Web网站基于HTTP协议，使用HTTP协议探测服务器上是否有内容更新，就必须频繁地让客户端请求服务器进行确认。

MQTT与HTTP

在浏览器中要实现这种效果，可以使用Comet技术，Comet是基于HTTP长连接的“服务器推”技术，主要有两种实现模型：基于AJAX的长轮询（long-polling）方式和基于Iframe及htmlfile的流（streaming）方式。如果要实现设备的反向控制，可能就要用到前面提到的Comet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。

当然，如果只是单纯地让设备定时上报数据而不做控制，也是可以使用HTTP协议的。

MQTT协议

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：

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；

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；

MQTT与XMPP

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；

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；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。

MQTT与CoAP

CoAP也是一个能与MQTT竞争的协议。其模仿HTTP的REST模型，服务端以URI方式创建资源，客户端可以通过GET、PUT、POST、DELETE方式访问这些资源，并且协议风格也和HTTP极为相似，例如一个设备有温度数据那么这个温度可以被描述为：

CoAP: //IP :5683/sensors/temperature

其中为设备的IP，5683为端口。

MQTT协议

MQTT与CoAP

不过，如果使用CoAP可能会让物联网后台的情况变得复杂，比如MQTT可以实现一个最简单的IoT架构：Device + MQTT服务器 + APP，手机端或Web端可以直接从MQTT服务器订阅想要的主题。而CoAP可能需要这样的架构：CoAP + Web + DataBase + App，使用CoAP必须经过DataBase才能转给第三方。

至于CoAP和MQTT孰优孰劣，这里不作定论。不过目前来说，CoAP资料还是略少。而且，MQTT除了可以应用于物联网领域，在手机消息推送、在线聊天等领域都可以有所作为。