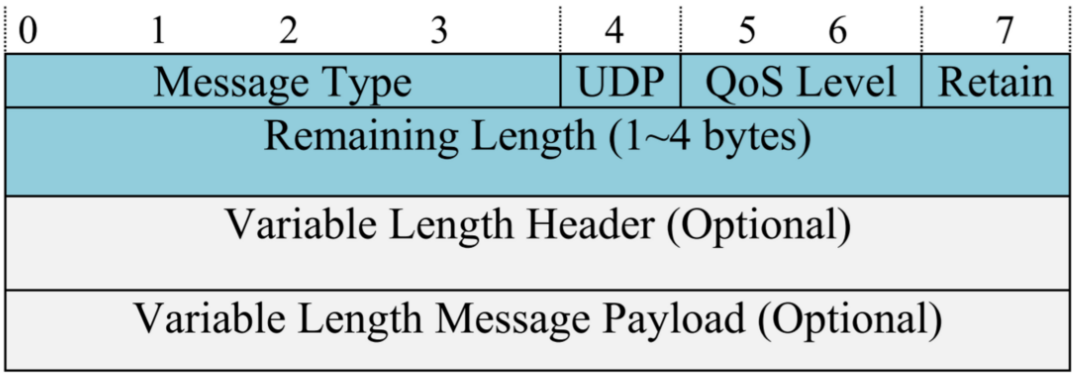
MQTT、CoAP 和 HTTP

协议栈：

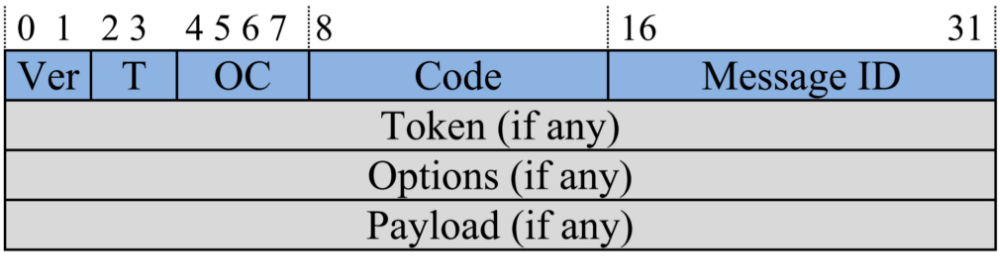
- MQTT：TCP长链接 | 发布订阅
- CoAP：UDP低功耗短链接 | request/response
- HTTP：TCP | request / respons

通讯消息格式

MQTT:



CoAP:



HTTP:

使用特点与场景

MQTT是多个客户端通过一个中央代理传递消息的**多对多**协议。它通过让客户端发布消息、代理决定消息路由和复制来解耦生产者和消费者。虽然MQTT持久性有一些支持，但它是最好的实时通讯总线。

CoAP基本上是一个在Client和Server之间传递状态信息的**单对单**协议。虽然它支持观察资源，但是CoAP最适合状态转移模型，而不是单纯的基于事件。

HTTP是适合使用在性能好一些的终端上，相对以上一些比较重，对设备要求相对高一些。不适合M2M的场景。

MQTT 特性

具有以下主要的几项特性：

1、使用发布/订阅消息模式，提供一对多的消息发布和应用程序之间的解耦；

2、消息传输不需要知道负载内容；

3、使用 TCP/IP 提供网络连接；

4、有三种消息发布的服务质量：

QoS 0：“最多一次”，消息发布完全依赖底层 TCP/IP 网络。分发的消息可能丢失或重复。例如，这个等级可用于环境传感器数据，单次的数据丢失没关系，因为不久后还会有第二次发送。

QoS 1：“至少一次”，确保消息可以到达，但消息可能会重复。

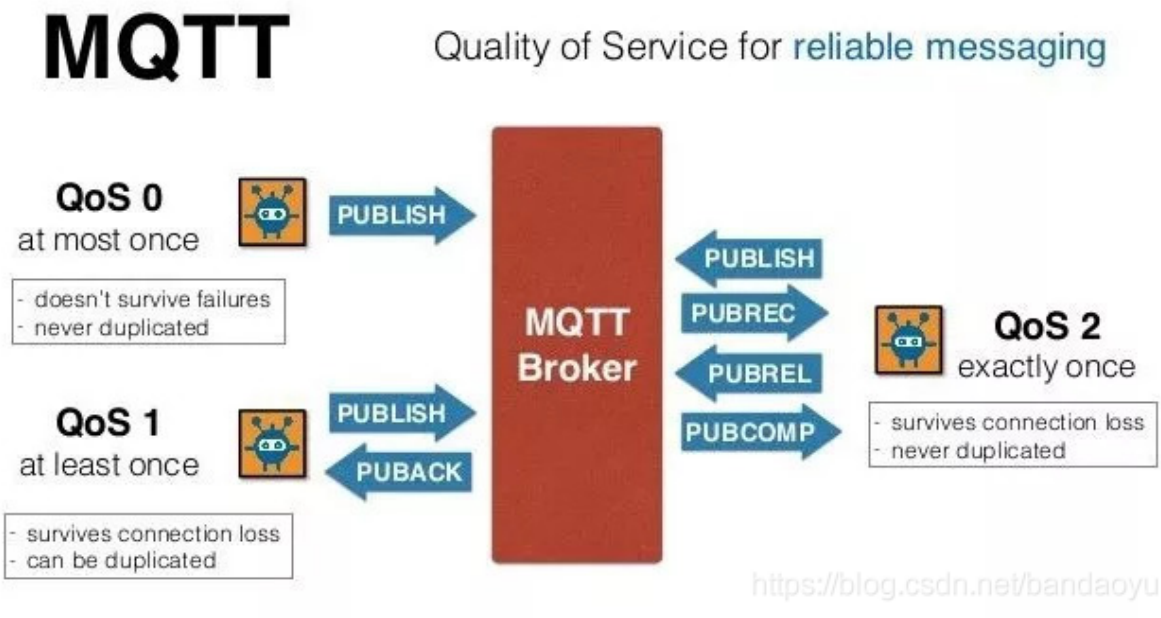
QoS 2：“只有一次”，确保消息只到达一次。例如，这个等级可用在一个计费系统中，这里如果消息重复或丢失会导致不正确的收费。

5、小型传输，开销很小（固定长度的头部是 2 字节），协议交换最小化，以降低网络流量；

6、使用 Last Will 和 Testament 特性通知有关各方客户端异常中断的机制；

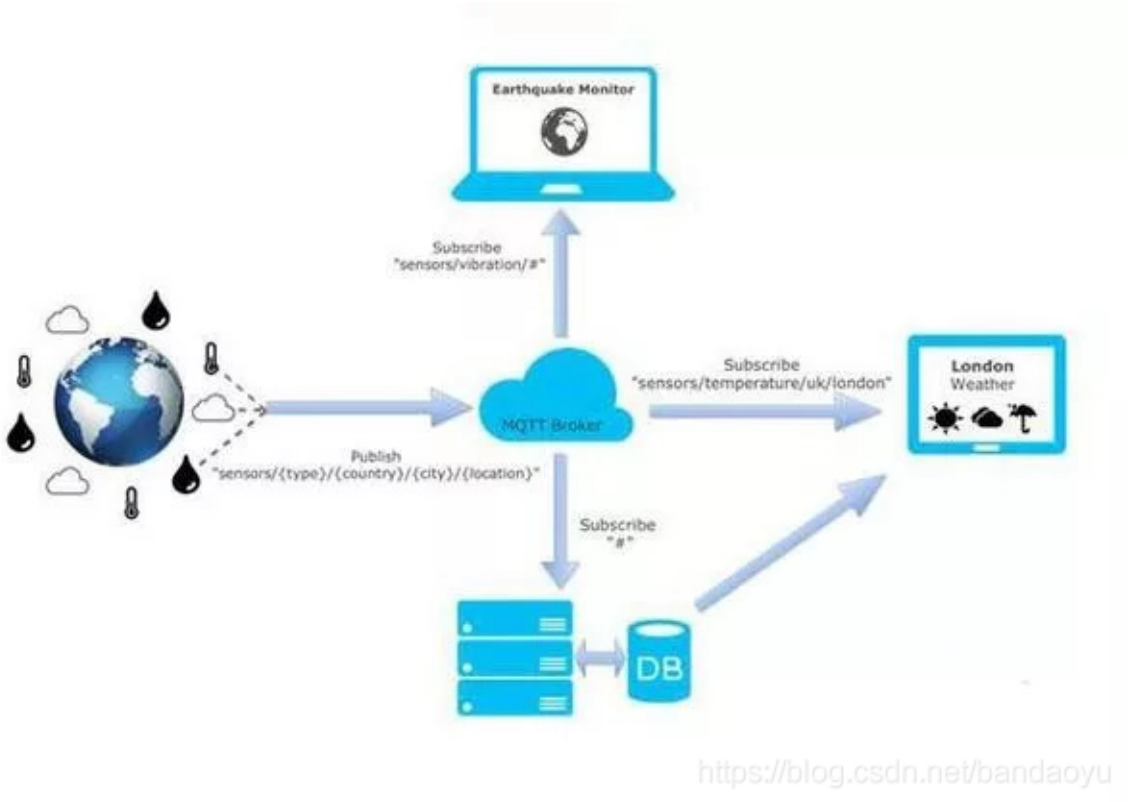
在MQTT协议中，一个MQTT数据包由：固定头（Fixed header）、 可变头（Variable header）、 消息体（payload）三部分构成。MQTT的传输格式非常精小，最小的数据包只有2个bit，且无应用消息头。

下图是MQTT为可靠传递消息的**三种消息发布服务质量**



发布/订阅模型允许MQTT客户端以一对一、一对多和多对一方式进行通讯。

下图是MQTT的发布 / 订阅消息模式



CoAP协议

CoAP是受限制的应用协议(Constrained Application Protocol)的代名词。

一种基于REST架构、传输层为UDP、网络层为6LowPAN（面向低功耗无线局域网的IPv6）的CoAP协议。

主要是一对一的协议

CoAP采用与HTTP协议相同的请求响应工作模式。CoAP协议共有4中不同的消息类型。

- CON——需要被确认的请求，如果CON请求被发送，那么对方必须做出响应。

- NON——不需要被确认的请求，如果NON请求被发送，那么对方不必做出回应。
- ACK——应答消息，接受到CON消息的响应。
- RST——复位消息，当接收者接受到的消息包含一个错误，接受者解析消息或者不再关心发送者发送的内容，那么复位消息将会被发送。

CoAP与HTTP的区别

MQTT协议是基于TCP，而CoAP协议是基于UDP。

1、MQTT协议不支持带有类型或者其它帮助Clients理解的标签信息，也就是说所有MQTT Clients必须要知道消息格式。而CoAP协议则相反，因为CoAP内置发现支持和内容协商，这样便能允许设备相互窥测以找到数据交换的方式。

2、MQTT是长连接而CoAP是无连接。MQTT Clients与Broker之间保持TCP长连接，这种情形在NAT环境中也不会产生问题。如果在NAT环境下使用CoAP的话，那就需要采取一些NAT穿透性手段。

3、MQTT是多个客户端通过中央代理进行消息传递的多对多协议。它主要通过让客户端发布消息、代理决定消息路由和复制来解耦消费者和生产者。MQTT就是相当于消息传递的实时通讯总线。CoAP基本上就是一个在Server和Client之间传递状态信息的单对单协议。

HTTP协议

http的全称是HyperText Transfer Protocol，超文本传输协议。

HTTP协议的两个过程，Request和Response，两个都有各自的语言格式：

请求报文格式：(注意这里有个换行)

响应报文格式：(注意这里有个换行)

方法method：

GET和POST方法等。

请求URL:

这里填写的URL是不包含IP地址或者域名的，是主机本地文件对应的目录地址，**所以我们一般看到的就是“/”。**

版本version：

状态码status: 404 502等

原因短语reason-phrase：

首部header：注意这里的header我们不是叫做头，而是叫做首部。可能有零个首部也可能有多个首部，每个首部包含一个名字后面跟着一个冒号，然后是一个可选的空格，接着是一个值，然后换行。

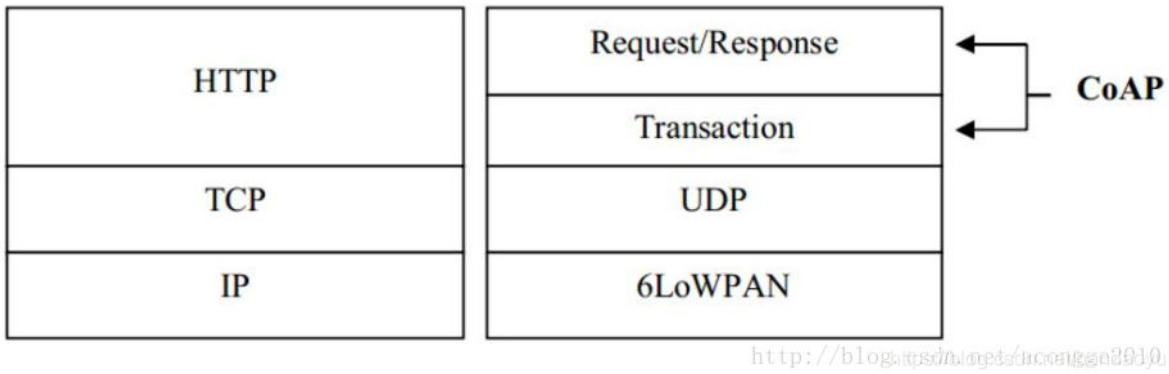
实体的主体部分entity-body：

实体的主体部分包含一个任意数据组成的数据块，并不是所有的报文都包含实体的主体部分，有时候只是一个空行加换行就结束了。

```
请求报文：
GET /index.html HTTP/1.1
Accept: text/*
Host: www.myweb.com
响应报文：
HTTP/1.1 200 OK
Content-type: text/plain
Content-length: 3
```

HTTP与CoAP的区别

CoAP是6LowPAN协议栈中的应用层协议，基于REST（表述性状态传递）架构风格，支持与REST进行交互。通常用户可以像使用HTTP协议一样用CoAP协议来访问物联网设备。而且CoAP消息格式使用简单的二进制格式，最小为4个字节。HTTP使用报文格式对于嵌入式设备来说需要传输数据太多，太重，不够灵活。



MQTT协议

物联网传输协议

1. 轻量级的发布/订阅式消息传输。
2. 为低带宽和不稳定的网络环境中的物联网设备提供可靠的网络服务。

它的核心设计思想是开源、可靠、轻巧、简单，具有以下主要的几项特性：

1. 非常小的通信开销（最小的消息大小为 2 字节）；
2. 支持各种流行编程语言（包括C，Java，Ruby，Python 等等）且易于使用的客户端；
3. 支持发布 / 预定模型，简化应用程序的开发；
4. 提供三种不同消息传递等级，让消息能按需到达目的地，适应在不稳定工作的网络传输需求

对于传统的HTTP和MQ协议，MQTT的优势在哪里呢?

表 1. 协议比较

标准	HTTP	MQ	MQTT
机密性	是	是	是
低协议开销	否	否	是
对不稳定网络的容忍	否	是	是
低功耗	否	否	是
数百万个连接的客户	否	否	是
推送通信	是*	是	是
客户端平台差异	是	否	是
防火墙容错	是	否	是

低协议开销
MQTT 的独特之处在于，它的每消息标题可以短至 2 个字节。MQ 和 HTTP 都拥有高得多的每消息开销。对于 HTTP，为每个新请求消息重新建立 HTTP 连接会导致重大的开销。MQ 和 MQTT 所使用的永久连接显著减少了这一开销。

对不稳定网络的容忍
MQTT 和 MQ 能够从断开等故障中恢复，而且没有进一步的代码需求。但是，HTTP 无法原生地实现此目的，需要客户端重试编码，这可能增加幂等性问题。

低功耗
MQTT 是专门针对低功耗目标而设计的。HTTP 的设计没有考虑此因素，因此增加了功耗。

数百万个连接的客户端
在 HTTP 堆栈上，维护数百万个并发连接，需要做许多的工作来提供支持。尽管可以实现此支持，但大多数商业产品都为处理这一数量级的永久连接而进行了优化。IBM 提供了 IBM MessageSight，这是一个单机架装载服务器，经过测试能处理多达 100 万个通过 MQTT 并发连接的设备。相反，MQ 不是为大量并发客户端而设计的。

推送通知
您需要能够及时地将通知传递给客户。为此，必须采用某种定期轮询或推送方法；从电池、系统负载和带宽角度讲，推送是最佳解决方案。

HTTP 只允许使用一种称为COMET 的方法，使用持久的 HTTP 请求来执行推送。从客户端和服务器的角度讲，此方法都很昂贵。MQ 和 MQTT 都支持推送，这是它们的一个基本特性。

客户端平台差异
HTTP 和 MQTT 客户端都已在大量平台上实现。MQTT 的简单性有助于以极少的精力在额外的客户端上实现 MQTT。

防火墙容错

一些企业防火墙将出站连接限制到一些已定义的端口。这些端口通常被限制为 HTTP（80 端口）、HTTPS（443 端口）等。HTTP 显然可以在这些情况下运行。MQTT 可封装在一个 WebSockets 连接中，显示为一个 HTTP 升级请求，从而允许在这些情况下运行。MQ 不允许采用这种模式。

事实上，MQTT的应用非常之广泛，几乎现在随便找一家大型的硬件、互联网企业，都可以找到MQTT的身影，例如Facebook、BP、alibaba、baidu等等

MQTT缺陷

由于MQTT本身的各项技术优势，越来越多的企业倾向于选用MQTT作为物联网产品通讯的标准协议，也因此，工程师们渐渐发现MQTT协议要想大规模商用，也有一些有待完善的功能。比如：

——没有齐备的SDK，不同的异构终端，需要有对应的与MQTT服务器通信的软件SDK包，比如MCU、Linux、Android、IOS、WEB等之间要实现互联互通必然需要不同的SDK包

——不支持File和AV，有些应用场景，需要传输的信息可能不仅仅限于指令，比如声音信号和视频信号，这些需要通过File和AV来实现通信。
——不支持与第三方HTTP的集成，虽然MQTT协议优于普通的HTTP协议，但是基于传统的HTTP协议的WEB服务器仍然占主流市场，那么这些服务器要实现与MQTT协议的互联互通，以降低升级成本也尤为关键。
——不支持负载均衡，为防止高并发和恶意攻击，负载均衡服务器也必不可少。
——不支持用户管理接口，用户在进行设备的行为数据分析的时候，显得尤为重要，这又是工业4.0、大数据时代的必然需求。
——不支持离线消息，弥补设备离线以后，MQTT服务器对设备的控制信息丢失的问题。
——不支持点对点通信，采用标准的MQTT协议，理论上可以通过相互订阅的方式实现点对点通信，但是逻辑相对复杂，并且对设备的安全性方面存在担忧。当设备B和设备C在同一主题的情况下，设备A无法知道是设备B还是设备C发送的消息，也有可能消息被设备D窃听。

——不支持群通信和群管理，实现了对群组成员的管理，群组成员之间能互通消息，这在一个设备被多人控制，或者多个设备被一人控制的这种场景下，尤为有用。