1.Abstract/概览（MQTT V3.1）

MQ遥测传输（MQTT）是轻量级的、基于代理的发布/订阅消息传输协议，此协议的设计开放、简单、轻量、易于实现。这些特点使得此协议非常适用于受限环境。例如，但不仅限于此：

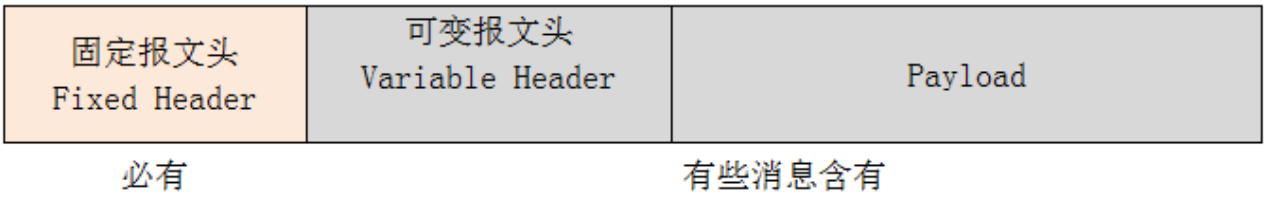
- 网络代价昂贵，带宽低、不可靠。
- 在嵌入设备中运行，处理器和内存资源有限。

该协议的特点包括：

- 使用发布/订阅消息模式，提供一对多的消息分发，解除了应用程序之间的耦合。
- 对负载内容屏蔽的消息传输。
- 使用TCP/IP提供基础的网络连接。
- 有三种消息传递服务质量：
  - “At most once”“至多一次”，消息发布完全依赖于底层的TCP/IP网络，会发生消息丢失或重复。这一级别可用于如下情况，如环境传感器数据，这种情况下，丢失一次读记录无所谓，因为第二个数据的发布紧跟其后。
  - At lease once“至少一次”，确保消息到达，但可能发生消息重复。
  - Exactly once“只有一次”，确保消息只到达一次。这一级别可用于如下情况，如计费系统中，消息重复或丢失会导致不正确的收费问题。
- 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。
- 提供一种机制，使得客户端异常中断时，能够使用 LastWill 和 Testament 特性通知有关各方。

2. Message format/消息格式

每个MQTT命令消息的消息头都包含一个固定的报头。有些消息需要一个可变的报头和一个payload。



下面将描述消息头的格式。

2.1. Fixed header/固定报头

每个MQTT命令消息的消息头都包含一个固定的报头。下表显示了固定的报头格式：

bit	7	6	5	4	3	2	1	0
byte 1	Message Type				DUP flag	QoS level		RETAIN
byte 2	Remaining Length							

Byte 1

包含Message Type（消息类型）和Flags（DUP，QoS级别，RETAIN）字段

Byte 2

（至少一个字节）包含Remaining　Length（剩余长度）字段。

这些字段将会在以下各部分说明。所有数据值都是按照bigi-endian顺序：高端字节跟在低端字节之前。

一个16位的字按照如下顺序：最高有效位（MSB）在前，最低有效位（LSB）在后。

2.1.1. Byte 1

2.1.1.1. Message Type 消息类型

位置：byte 1，bits 7-4

表现为一个4-bit的无符号值，这个版本协议的枚举值如下表所示：

Mnemonic	Enumeration	Description
Reserved	0	Reserved
CONNECT	1	Client request to connect to Server
CONNACK	2	Connect Acknowledgment
PUBLISH	3	Publish message
PUBACK	4	Publish Acknowledgment
PUBREC	5	Publish Received (assured delivery part 1)
PUBREL	6	Publish Release (assured delivery part 2)
PUBCOMP	7	Publish Complete (assured delivery part 3)
SUBSCRIBE	8	Client Subscribe request
SUBACK	9	Subscribe Acknowledgment
UNSUBSCRIBE	10	Client Unsubscribe request
UNSUBACK	11	Unsubscribe Acknowledgment
PINGREQ	12	PING Request
PINGRESP	13	PING Response
DISCONNECT	14	Client is Disconnecting
Reserved	15	Reserved

0->保留；

1->连接请求：客户端请求连接到服务器；

2->连接确认；

3->发布消息；

4->发布确认；

5->发布信息收到（确保分发的第1部分）；

6->发布信息分发（确保分发的第2部分）；

7->发布完成　　（确保分发的第3部分）；

8->客户端订阅请求；

9->订阅确认；

10->客户端取消订阅请求；

11->取消订阅确认；

12->ping请求；

13->ping响应；

14->客户端正在断开连接；

15->保留；

2.1.1.2. FLAGS（DUP、QoS、Retain）

byte1剩余的bits包含如下的字段：DUP，QoS及RETAIN，bit位置如下表所示

Bit position	Name	Description
3	DUP	Duplicate delivery
2-1	QoS	Quality of Service
0	RETAIN	RETAIN flag

DUP

位置：byte 1，bit 3

当client或者server试图重发“PUBLISH”、“PUBREL”、“SUBSCRIBE”或“UNSUBSCRIBE”消息时，这个flag会被设置。这个flag适用于QoS值大于零（0），且需要确认的消息。当DUP位被设置时，可变头部包含一个消息ID。

消息接收者应将这个Flag视作为一个提示，那就是这个消息此前可能已经被接收过。但是检测消息是否重复时不应该依赖此flag。

QoS

位置：byte 1，bits 2-1；

此flag标明了发布消息的交付质量等级。QoS级别如下表所示：

QoS value	bit 2	bit 1	Description		
0	0	0	At most once	Fire and Forget	<=1
1	0	1	At least once	Acknowledged delivery	>=1
2	1	0	Exactly once	Assured delivery	=1
3	1	1	Reserved		

RETAIN

位置：byte 1，bit 0

这个标识只会用于“PUBLISH”消息。当客户端发布一个消息到服务器端时，如果Retain flag被设为1，则服务器在将消息分发给当前的订阅者后应该保留这条消息（类似state类型的发布）。

当一个新的订阅者订阅某一个主题，如果Retain flag被设置，则此主题最后保留的消息会被发送给此订阅者。如果没有任何保留的消息，则不会发送。

这对于发布者基于“report by excepion”发布消息时非常有用，这种情况下，两个消息之间可能会有一定的间隔。这样，新的订阅者会立即接收保留的数据，或者目前为止最好的结果数据。

如果初始PUBLISH到达时，某个订阅者已经存在，那么在服务器给这个订阅者发布一个PUBLISH时，Retain标识不应该被重置，不管初始PUBLISH的Retain flag是什么。这样，客户端就可以区分正在接收的消息是保留的消息还是刚刚发布的消息。（类似状态消息支持先订阅后发布，如果是先有订阅者，后有发布者时，Retain不会被设置，因为此订阅者收到的不是保留消息）

保留的消息在服务器重启后应该继续保留（持久保存保留消息）。

如果服务器收到了一个payload长度为0且Retain flag设置在某一个主题上的消息，则服务器可以删除此主题的保留消息。（状态类发布的消息可被删除，收到消息体长度为0，Retain为1的某个主题的消息时，Server会删除此topic的保留消息）

2.1.2. Byte2

2.1.2.1. Remaining Length

位置：byte 2

表示当前消息剩余的字节数，包括可变报文头以及payload。

对于长度多达127个字节的消息，可变长度的编码方案用了一个单独的字节。对于较长的消息处理方式如下。每个字节的7位用于编码Remaining Length数据，第8位表示在下面还有值。每个字节编码128个值和一个“延续位”。例如，数字64，编码为一个字节，十进制表示为64，十六进制表示为0×40。数字321（=65+2*128）编码为两个字节，重要性最低的放在前面，第一个字节为65+128=193。注意：如果最高位被设置，则表明后续至少还有一个字节。第二个字节是2。

协议限制表示所用的字节数最大只能到四个字节，这样，应用程序所能发送的消息长度最长可达268435455（256M）。

一系列的数字分别表示为：0xFF，0xFF，0xFF，0x7F。

下表显示了增加字节数后，Remaining Length所表示的值：

Digits	From	To
1	0 (0x00)	127 (0x7F)
2	128 (0x80, 0x01)	16 383 (0xFF, 0x7F)
3	16 384 (0x80, 0x80, 0x01)	2 097 151 (0xFF, 0xFF, 0x7F)
4	2 097 152 (0x80, 0x80, 0x80, 0x01)	268 435 455 (0xFF, 0xFF, 0xFF, 0x7F)

分别表示：　1个字节时，从0到127；

2个字节时，表示从128到16383；

3个字节时，表示从16384到2097151；

4个字节时，表示从2097152到268435455.

将十进制数字按照可变长度编码规范进行编码的算法如下：

```
do
```

```
digit = X MOD 128
X = X DIV 128
// if there are more digits to encode, set the top bit of this digit
if ( X > 0 )
digit = digit OR 0x80
endif
'output' digit
while ( X> 0 )
```

MOD在C语言中是模运算符号，DIV是整数除（/ in C），OR是位或操作（| in C）.

对Remaining Length域进行解码的算法如下：

```
multiplier = 1
value = 0
do
digit = 'next digit from stream'
value += (digit AND 127) * multiplier
multiplier *= 128
while ((digit AND 128) != 0)
```

AND是位与操作（& in C).

当算法结束时，value以字节为单位包含了Remaining Length。

Remaining Length编码不是可变报文头的一部分。编码Remaining Length所用的字节数不会添加到Remaining Length的值中。可变长度”扩展字节（extension bytes）”是固定报文头的一部分，而不是可变报文头的一部分。

2.2. variable header/可变报文头

某些类型的MQTT命令消息也包括一个variable header组件，位于fixed header和payload之间。

Remaining Length字段的可变长度不是variable header的一部分，Remaining Length值的字节数没有把Remaining Length字段的字节计算在内。Remaining Length的值只考虑variable header和playload。更多信息查看Fixed header。

Variable header各个字段的格式在下面的章节，按照每个字段在报文头中必须出现的**顺序**进行描述。

2.2.1. Protocol name 协议名

Protocol name（协议名称）出现在MQTT CONNECT消息的variable header中。该字段是UTF8

编码的字符串，代表协议名称MQIsdp，大写显示。

2.2.2. Protocol Version 协议版本

协议版本表示在CONNECT消息的variable header中。

这个字段是一个8位的无符号值，表示客服端所使用的协议的修订级别，当前版本的协议为3（0×03），如表所示：

bit	7	6	5	4	3	2	1	0
	Protocol Version							
	0	0	0	0	0	0	1	1

2.2.3. Connect flags

Clean session， Will， Will QoS 和Retain flags都会在CONNECT消息的可变报文头中体现。

Clean session flag

Position：Connect flags字节的第1位

如果不设置（0），那么，客户端订阅者断开连接后，服务器必须存储客户端的订阅信息。还包括对订阅的主题所对应的QoS1和QoS2消息的持续存储 ,这样 ,当客户端重新连接时能够把消息分发给客户端。服务器还必须保持在连接断开时正在分发中的消息的状态，这些信息必须一直保持，知道客户端重新建立连接。（类似持久订阅）

如果设置(1),那么服务器必须丢弃任何之前维护的关于客户端的信息 ,并将这个连接看作“clean”。当客户端断开连接时，服务器还必须丢弃任何状态。（类似非持久订阅）

通常情况下，一个客户端要么采用这个模式，要么采用另外一个模式，且不会改变，该选择将取决于应用。客户端如果采用clean会话，将不会接收到过期的信息，每次连接时必须重新订阅。客户端如果采用non-clean会话，将不会错过任何在连接断开时所发布的QoS 1或者QoS 2的信息。QoS 0的消息永远不会被存储，因为这类消息是基于最佳条件发布的。（意思是：发布QoS 0的消息时，发布相关的条件很

好）

这个标志以前被称为“Clean start”，现在已经被重命名，因为事实上它应用于整个会话，而不仅仅是应用于最初的连接阶段。

服务器可以为客户端提供一种管理机制，用于清除所存储的那些永远都不会重连的客户端的信息。（由ClientID区分每个客户端）

bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
	x	x	x	x	x	x		x

这个字节的Bit 0 of在这个协议的当前版本中没有用到，为以后版本保留。

Will flag

位置： Connect flags字节的第2位

当服务器与客户端通信时遇到I/O错误或客户端没有在Keep Alive期内进行通讯时，Server会代表客户端发布一个Will消息。服务器收到来自客户端的DISCONNECT消息时，并不会触发Will消息的发送。

如果设置了Will flag，Connect flags的字节中就必须有Will QoS和Will Retain字段，payload中必须有Will Topic和Will message 字段。（可以用于监控客户端与服务器之间的连接状况）

Will flag的格式如下表所示：

bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
	x	x	x	x	x		x	x

这个字节的Bit 0在此协议的当前版本中没有被用到，为将来预留。

Will QoS

位置: Connect flags字节的第4、3位

连接中的客户端如果在Will QoS字段中为Will消息指定了QoS级别，一旦发生非主动的连接断开，Will消息将会被发送。Will消息定义在CONNECT消息的payload中。

如果设置了Will flag，那么Will QoS字段是强制性的，否则Will QoS字段的值将被忽略。

Will QoS的值是0（0×00），1（0×01），or 2（0×02）。Will QoS标志显示在下面的表格中。

bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
	x	x	x			1	x	x

Will Retain flag

位置： connect flags字节的第5位

Will Retain flag指明了服务器是否应该保留Will消息，此Will消息是在客户端意外断开连接时由服务器代表客户端所发布的。

如果设置Will flag，则Will Retain flag是强制性的，否则Will Retain flag将被忽略。

Will Retain flag的格式如下表所示:

bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
	x	x		x	x	1	x	x

Bit 0本版本不用，为以后预留.

User name and password flags

Position: Connect flags字节的第6、7位

连接中的客户端可以指定用户名和密码，设置这两个标志位意味着一个用户和一个可选的密码会包含在CONNECT消息的payload里。

如果设置了User Name标志，则User Name字段是强制性的，否则User Name的值将被忽略。如果设置Password标志，则Password字段是强制性的，否则Password的值将被忽略。如果没有设置User Name，而提供Password是无效的。仅仅提供了password而没有提供user name是无效的。

bit	7	6	5	4	3	2	1	0
	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
			x	x	x	x	x	x

2.2.4. Keep Alive timer

Keep Alive timer位于MQTT CONNECT消息的可变报文头（variable header）中。

Keep Alive timer为秒级，定义了客户端接收消息时消息之间的最大时间间隔。这样，服务器无需等待很长的TCP/IP超时时间即可发现与客户端的连接已经断了。

客户端有义务在每一个Keep Alive time内发生一个消息. 在这个时间周期内，如果没有业务数据相关（data-related）的消息，客户端会发一个PINGREQ，相应的，服务器会返回一个PINGRESP消息进行确认。（类似心跳机制）

如果服务器在一个半（1.5）的Keep Alive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，就像客户端发送了一个DISCONNECT消息一样。断开连接不会影响这个客户端的任何订阅。具体可参照DISCONNECT。

如果客户端在发送了PINGREQ消息后的一个Keep Alive周期内没有收到PINGRESP消息应答，应该关闭TCP/IP socket连接。

Keep Alive timer用一个16位的值表示这个时间周期（单位为秒）。实际值是由应用指定的，通常设置成几分钟。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。（超过此时间无消息，连接断开）

Keep Alive timer的格式见下表。Keep Alive Timer的两个字节是按照先MSB，后LSB的顺序（big-endian）。

bit	7	6	5	4	3	2	1	0
	Keep Alive MSB							
	Keep Alive LSB							

2.2.5. Connect return code

Connect return code是在CONNACK消息的可变报文头（variable header）. 这个字段定义了一个

无符号的byte返回码。下表显示了值的含义与针对的消息类型。返回值0通常表示成功。

Enumeration	HEX	Meaning
0	0x00	Connection Accepted
1	0x01	Connection Refused: unacceptable protocol version
2	0x02	Connection Refused: identifier rejected
3	0x03	Connection Refused: server unavailable
4	0x04	Connection Refused: bad user name or password
5	0x05	Connection Refused: not authorized
6-255		Reserved for future use

bit	7	6	5	4	3	2	1	0
	Return Code							

2.2.6. Topic name

主题名位于MQTT PUBLISH消息的可变报文头（variable header）中。

主题名是一把钥匙，指明了payload数据会被发布到哪个信息通道。订阅者使用这把钥匙来确定他们想接收发布消息的信息通道。

主题名是UTF-8编码的字符串。要获得更多信息，请参见“MQTT和UTF-8”一节。主题名最多可以包含**32,767**个字符。

2.3. Payload

下列几种类型的MQTT命令消息含有payload：

CONNECT:

Payload包含一个或多个UTF-8编码的字符串，为客户端指定了一个唯一的标识符（ClientID），一个Will主题和消息（Will topic/Will Message）、所用的User Name和Password。除了ClientID，其他都是可选的，是否含有可选内容是由可变报文头中的标识（flags）决定的。

SUBSCRIBE

payload包含一系列客户端可以订阅的主题名（topic names）、QoS等级。这些字符串都是UTF编码。

SUBACK

Payload包含一系列授权的QoS级别。利用这些QoS级别，服务器的管理员可以授权客户端订阅某个特定的主题。授权的QoS级别与SUBSCRIBE消息中的topic名的排列顺序是一致的。

PUBLISH消息中的payload部分只包含特定于应用的数据。没有对数据的特性和内容做任何假设，消息的这部分内容被看作是一个BLOB。（意思就是不会对所发布的消息内容进行任何处理）

如果想让应用压缩payload数据，则需要在应用中定义适当的payload flag字段（如tlq的compressflag）来处理压缩的详细工作。但不能在固定或者可变报文头里定义任何特定于应用的标志。（压缩是针对应用数据的，不是针对协议的，所以应用必须自己处理压缩）

2.4. Message Identifier

在下面的MQTT消息的可变报文头中会有Message Identifier（消息标识）：PUBLISH，PUBACK，PUBREC，PUBREL，PUBCOMP，SUBSCRIBE，SUBACK，UNSUBSCRIBE，UNSUBACK。

消息标识字段仅会出现在QoS级别为1或2的消息中（Fixed header中的QoS位会表明QoS的级别）。更多信息，请参见Quality of Service levels and flows。

消息标识符是一个16位的无符号整数，在某个特定方向的通信中，一系列正在传输中（in-flight）的消息必须保证每个消息的Message ID唯一。通常情况下，消息标识严格按照消息传输的顺序增加（类似序数增加），但是并不要求这样做。

客户端会维护自己的消息标识列表，所连接的服务器也会使用这些消息标识。客户端很有可能发送一个Message ID为1的PUBLISH消息，同时收到了一个MessageID为1的PUBLISH消息。Message Identifier的两个字节的顺序是先MSB，然后LSB（big-endian）

注意：不要将Message ID设为0。0是作为无效消息ID保留的。

bit	7	6	5	4	3	2	1	0
	Message Identifier MSB							
	Message Identifier LSB							

2.5. MQTT and UTF-8

UTF-8是一个有效的对Unicode字符串的编码，优化了对ASCII字符的编码，为基于文本（text-based）的通信提供支持。

在MQTT中，字符串的前面都会有两个字节，用来表明字符串的长度，如下表所示：

bit	7	6	5	4	3	2	1	0
byte 1	String Length MSB							
byte 2	String Length LSB							
bytes 3 ...	Encoded Character Data							

字符串长度是编码后的字符串的字节数，而不是字符的个数，例如，字符串OTWP用UTF-8进行编码，如下表所示：

bit	7	6	5	4	3	2	1	0
byte 1	Message Length MSB (0x00)							
	0	0	0	0	0	0	0	0
byte 2	Message Length LSB (0x04)							
	0	0	0	0	0	1	0	0
byte 3	'O' (0x4F)							
	0	1	0	0	1	1	1	1
byte 4	'T' (0x54)							
	0	1	0	1	0	1	0	0
byte 5	'W' (0x57)							
	0	1	0	1	0	1	1	1
byte 6	'P' (0x50)							
	0	1	0	1	0	0	0	0

Java中， `writeUTF()` 和 `readUTF()` 数据流方法用这种格式。

2.6. Unused bits

任何被标记为 Unused 的位都应该被设为 zero（0）。

3.Command messages

3.1.CONNECT- 客户端向服务器请求建立连接

当客户端向服务器建立一个 TCP/IP 连接时，协议层会话必须使用一个 CONNECT flow 建立。

3.1.1. Fixed header

fixed header 的格式见下表：

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (1)				DUP flag	QoS level		RETAIN
	0	0	0	1	x	x	x	x
byte 2	Remaining Length							

DUP, QoS, 和RETAIN flags没有在CONNECT消息中使用.

Remaining Length是variable header (12 bytes)的长度和Payload的长度. 这可以是一个多字节的字段.

3.1.2. Variable header

一个 variable header 格式的例子见下表：

	Description	7	6	5	4	3	2	1	0
Protocol Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (6)	0	0	0	0	0	1	1	0
byte 3	'M'	0	1	0	0	1	1	0	1
byte 4	'Q'	0	1	0	1	0	0	0	1
byte 5	'I'	0	1	0	0	1	0	0	1
byte 6	's'	0	1	1	1	0	0	1	1
byte 7	'd'	0	1	1	0	0	1	0	0
byte 8	'p'	0	1	1	1	0	0	0	0
Protocol Version Number									

	Description	7	6	5	4	3	2	1	0
byte 9	Version (3)	0	0	0	0	0	0	1	1
Connect Flags									
byte 10	User name flag (1) Password flag (1) Will RETAIN (0) Will QoS (01) Will flag (1) Clean Session (1)	1	1	0	0	1	1	1	x
Keep Alive timer									
byte 11	Keep Alive MSB (0)	0	0	0	0	0	0	0	0
byte 12	Keep Alive LSB (10)	0	0	0	0	1	0	1	0

User name flag
Set (1).
Password flag
Set (1).
Clean Session flag
Set (1).
Keep Alive timer
Set to 10 seconds (0x000A).
Will message
Will flag is set (1)
Will QoS field is 1
Will RETAIN flag is clear (0)

3.1.3. Payload

CONNECT消息的payload包含一个或多个的UTF-8编码的字符串，这是基于variable header中的标识（flags）。如果有字符串存在，则必须按下列顺序出现:

Client Identifier

第一个UTF编码的字符串。客户端标识符（Client ID）的长度介于1到23个字符，唯一标识连接到服务器的客户端。连接到某一台服务器上的所有客户端的Client Identifier必须是唯一的，这是处理QoS级别为1和2的带消息ID的消息的关键。如果客户端ID的字符数超过23个，服务器对这个CONNECT消息的响应是一个CONNACK返回码2：Identifier Rejected。

Will Topic

如果设置了Will Flag，这是下一个UTF-8编码的字符串。Will Message会被发布给Will

Topic。QoS级别通过Will QoS字段来定义，RETAIN状态通过Will RETAIN标识定义在variable header里。

Will Message

如果设置了Will Flag，这是下一个UTF编码的字符串。Will Message定义了客户端意外断开时会发布到Will Topic的消息内容，这可以是一个长度为0的消息。

虽然在CONNECT消息中的Will Message是UTF编码的，当Will Message发布给Will Topic时，只有消息的字节会被发送，前两个长度字节不会被发送。因此，该消息必须只能由7位ASCII字符组成。

User Name

如果设置了User Name标识，这是下一个UTF-8编码的字符串。User name标识了正在连接的用户的名字，这个名字可用于身份验证。建议用户名的长度等于或小于12个，但这不是必须的。

请注意：为了兼容原来的MQTT V3规范，Fixed Header中的Remaining Length字段优先于User Name标识。服务器实施必须允许这样一种可能性：设置了User Name标识，但是没有设置User Name字符串。这是有效的，应允许连接继续建立。

Password

如果设置了Password标识，这是下一个UTF编码的字符串。正在连接的用户对应的Password可用于身份验证。建议密码长度为12或更少，但这不是必须的。

请注意：为了与兼容原来的MQTT V3规范，Fixed Header中的Remaining Length字段优先于Password标识。服务器必须允许这样一种可能性：设置了Password标识，但是没有设置Password字符串。这是有效的，应允许连接继续建立。

Response

服务器发送一个CONNACK消息作为客户端发送的CONNECT消息的响应。如果服务器在TCP/IP连接建立后在一个合理的时间内没有收到一个CONNECT消息，服务器应关闭连接。

如果客户端在一个合理的时间内没有收到一个来自服务器的CONNACK消息，客户端应该关闭TCP/IP socket连接，并通过重开一个到服务端的socket并发出一个CONNECT消息的方式重启

一个session。

（意思是：在socket建立后，如果客户端和服务器双方没有在合理时间内收到连接相关的消息，则都会主动断开连接）

在这两种场景中，一个“合理”的时间量取决于应用和通信基础设施。

如果一个具有相同Client ID的客户端已经连接到服务器，在完成新客户端的CONNECT流程之前，服务器断开与“older”客户端的连接。

如果客户端发送一个无效的CONNECT消息，服务器应当关闭这条连接。如CONNECT消息提供了无效的协议名称或协议版本号。

如果服务器可以充分解析CONNECT消息，以确定被请求了一个无效的协议，服务器可能会在断开连接前尝试发送CONNACK，CONNACK包含“Connection Refused: unacceptable protocol version”。

3.2. CONNACK – 连接请求的确认

CONNACK message是服务器为了响应来自客户端的CONNECT请求而发送的消息.

3.2.1. Fixed header

fixed header 格式见下表：

bit	7	6	5	4	3	2	1	0
byte 1	Message type (2)				DUP flag	QoS flags		RETAIN
	0	0	1	0	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

DUP, QoS和RETAIN flags都没有用在CONNACK message中。

3.2.2. Variable header

variable header 格式见下表：

	Description	7	6	5	4	3	2	1	0
Topic Name Compression Response									
byte 1	Reserved values. Not used.	x	x	x	x	x	x	x	x
Connect Return Code									

	Description	7	6	5	4	3	2	1	0
byte 2	Return Code								

这个无符号的单字节的连接返回码字段的值见下表：

Enumeration	HEX	Meaning
0	0x00	Connection Accepted
1	0x01	Connection Refused: unacceptable protocol version
2	0x02	Connection Refused: identifier rejected
3	0x03	Connection Refused: server unavailable
4	0x04	Connection Refused: bad user name or password
5	0x05	Connection Refused: not authorized
6-255		Reserved for future use

如果唯一的客户标识的长度不是1-23，则会返回值为2的返回码（identifier rejected）

3.2.3. Payload

没有payload.

3.3. PUBLISH-发布消息

为向感兴趣的订阅者分发消息，客户端向服务器发布了一条PUBLISH消息。每一条PUBLISH消息都与一个主题名相关联（主题名又称Subject或Channel）。这是一个分层的名称空间，定义了一个信息源分类，订阅者可以注册一个兴趣。一个发布到某个特定主题名的消息会被传递给订阅了该主题的在线订阅者。

如果客户端订阅了一个或多个主题，任何发布到这些主题上的消息都会作为 PUBLISH 消息被服务器发送给客户端。

3.3.1. Fixed header

格式见下表

bit	7	6	5	4	3	2	1	0
byte 1	Message type (3)				DUP flag	QoS level		RETAIN
	0	0	1	1	0	0	1	0

bit	7	6	5	4	3	2	1	0
byte 2	Remaining Length							

QoS level

设为 1.　　See QoS for more details

DUP flag

设为 zero (0). 意味着这个消息是第一次被发送。 See DUP for more details.

RETAIN flag

设为 zero. 意味着不保留.　　See Retain for more details.

Remaining Length field

variable header 长度加上 payload 的长度. 可以是一个多字节的字段.

3.3.2. Variable header

variable header包含下面的字段:

Topic name

一个UTF-编码的字符串.

一定不能含有Topic模糊匹配字符，当客户端用模糊匹配符订阅时，收到的消息中，这个字符串将会是完整的topic，为发布者最初发布时指定的topic，而不是客户端的订阅串。

Message ID

在QoS level为1 and 2的消息中才有此项. See Message identifiers for more details.

下表是一个 PUBLISH 消息的 variable header 的例子:

Field	Value
Topic Name:	"a/b"
QoS level	1
Message ID:	10

这个例子中，variable header 的格式见下表：

	Description	7	6	5	4	3	2	1	0
Topic Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	'a' (0x61)	0	1	1	0	0	0	0	1
byte 4	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 5	'b' (0x62)	0	1	1	0	0	0	1	0
Message Identifier									
byte 6	Message ID MSB (0)	0	0	0	0	0	0	0	0
byte 7	Message ID LSB (10)	0	0	0	0	1	0	1	0

Payload

包含发布的数据。数据的格式和内容由应用决定。fixed header中的Remaining Length字段包括variable header的长度和payload的长度。因此，发布含有0长度payload的PUBLISH是有效的。

Response

PUBLISH消息的响应依赖于QoS的级别。下表显示了预期的Response。

QoS Level	Expected response
QoS 0	None
QoS 1	PUBACK
QoS 2	PUBREC

Actions

Publisher消息可以从发布者发送到服务器，或从服务器发布到订阅者。当接收者收到消息时，其行动依赖于消息的QoS级别：

QoS 0

消息对任何感兴趣的一方都可见。

QoS 1

将消息持久化，使其对任何感兴趣的一方都可见，并返回一个PUBACK给发送方。

QoS 2

将消息持久化，不使其对任何感兴趣的一方都可见，并返回一个PUBREC消息给发送方。

如果服务器收到这个消息，感兴趣的各方是指订阅了这个PUBLISH消息的主题的订阅者。如果订阅者收到这个消息，感兴趣的各方是指订阅了一个或多个主题的客户端应用，这个客户端应用正等待来自服务器的消息。

具体请参照 Quality of Service levels and flows

请注意：如果服务器的实现中并没有授权一个来自客户端的PUBLISH，那就无法通知那个客户端。因此，服务器必须按照正常的QoS的规则做出肯定的确认，客户端将不会被告知，它未被授权去发布这个消息。

3.4. PUBACK-发布确认

PUBACK消息是对QoS级别为1的PUBLISH消息的响应。一个PUBACK消息可以是服务器为了响应来自发布客户端的PUBLISH消息而发送，也可以是订阅者为了响应来自服务器的PUBLISH消息而发送的。

3.4.1. Fixed header

fixed header的格式见下表：

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (4)				DUP flag	QoS level		RETAIN
	0	1	0	0	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

QoS level

Not used.

DUP flag

Not used.

RETAIN flag

Not used.

Remaining Length field

这是variable header的长度（2 bytes）. 可以是多字节字段。

3.4.2. Variable header

包含正在被确认的PUBLISH消息的消息标识符（Message ID）。下表显示了variable header的格式：

bit	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

3.4.3. Payload

没有payload.

3.4.4. Actions

当客户端收到PUBACK消息，就会丢弃原来的消息，因为服务器也收到了这个消息（且做了日志）。

3.5. PUBREC–确保发布被收到

PUBREC消息是用来响应QoS级别为2的PUBLISH消息的。这是QoS级别为2的protocol flow（协议流）的第二个消息。PUBREC消息由服务器发送以响应发布客户端的PUBLISH消息，或者由订阅者发送以响应来自服务器的PUBLISH消息。
（QoS级别为2时，当客户端发布一条消息，服务端收到后要给出PUBREC响应；订阅者收到发布的消息也要给出PUBREC响应，确保发布过程的各个环节）

3.5.1. Fixed header

下表为 Fixed header 的格式：

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (5)				DUP flag	QoS level		RETAIN
	0	1	0	1	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

QoS level
没有用到
DUP flag
没有用到.

RETAIN flag

没有用到

Remaining Length field

variable header 的长度（2 bytes）. 可以是一个多字节的字段.

3.5.2. Variable header

variable header包含了被确认的PUBLISH消息的消息标识（Message ID）

下表为variable header的格式：

bit	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

3.5.3. Payload

没有payload.

3.5.4. Actions

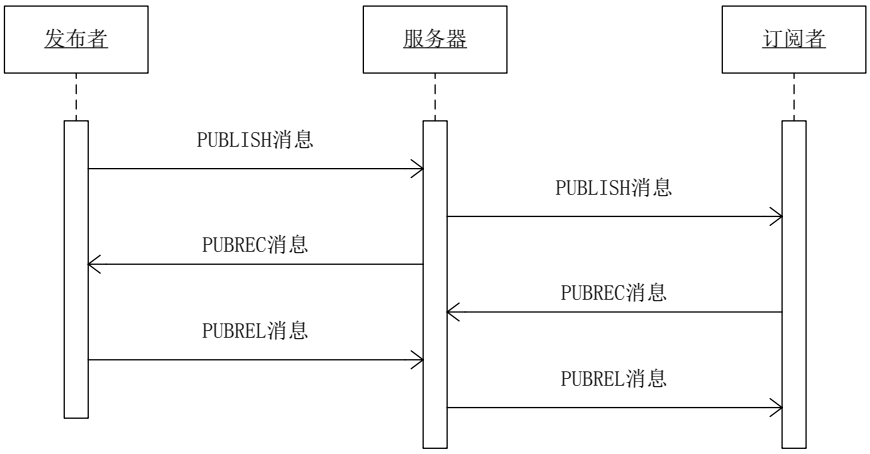
当收到一个PUBREC消息，接收者会使用与PUBREC消息相同的MessageID发送一个PUBREL消息给发送者。（PUBREC的接收者用PUBREL做出回应，表示自己收到了PUBREC，且二者使用相同的消息标识）。

3.6. PUBREL-Assured Publish Release

PUBREL消息是QoS级别为2的协议流的第3个消息。

PUBREL消息用用来响应PUBREC消息的，可以是发布者对来自服务器的PUBREC的响应，也可以是服务器对来自订阅者的PUBREC的响应。

（双方互相确认，确保发布的消息到了server，也到了各个订阅者）



3.6.1. Fixed header

fixed header 格式见下表.

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (6)				DUP flag	QoS level		RETAIN
	0	1	1	0	0	0	1	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

QoS level

PUBREL消息的QoS级别为1，会以PUBCOMP格式作为确认形式。重试的处理流程跟PUBLISH消息一样。（一旦失败，就需要重试）

DUP flag

设为zero (0). 这就意味着这个消息是第一次被发送. 向了解更多请参见DUP.

RETAIN flag

没有用到

Remaining Length field

variable header 的长度 (2 bytes). 可以是一个多字节的字段.

3.6.2. Variable header

variable header包含一个跟正在确认的PUBREC消息相同的Message ID 。 variable header格式如下表：

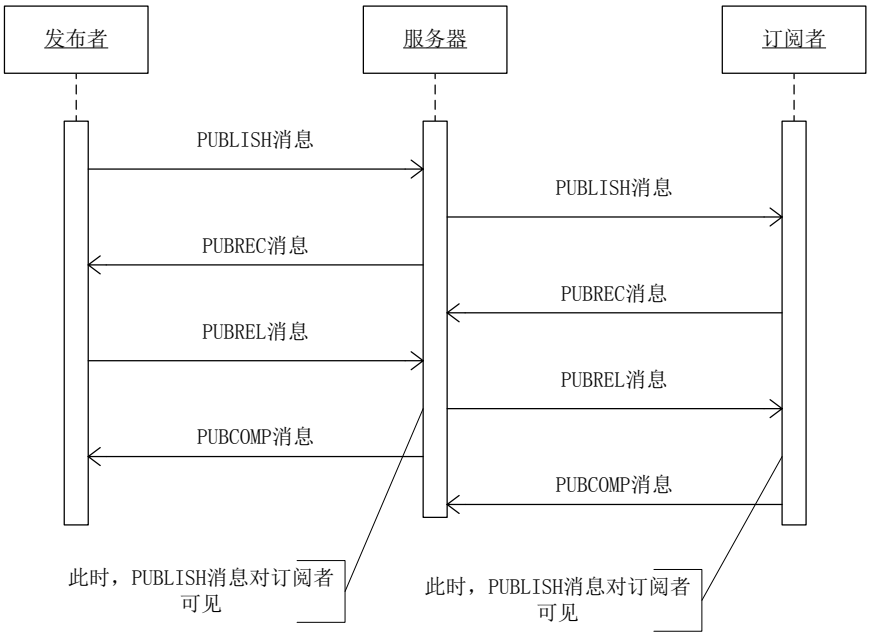
bit	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

3.6.3. Payload

无payload.

3.6.4. Actions

当服务器接收来自发布者的PUBREL消息后，才会使原消息(PUBLISH消息)对感兴趣的订阅者可见，并发送一个拥有相同Message ID的PUBCOMP消息给发布者。当订阅者收到来自服务器的PUBREL的消息，才会使得消息对订阅应用程序可见，并发送一个PUBCOMP消息给服务器。



3.7. PUBCOMP-确保发布完成

这是QoS基本为2的协议流的第4个，也是最后一个消息。

这个消息是对PUBREL消息的响应。

3.7.1. Fixed header

fixed header 格式见下表：

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (7)				DUP flag	QoS level		RETAIN

bit	7	6	5	4	3	2	1	0
	0	1	1	1	x	x	x	x
byte 2	Remaining Length (2)							
	0	0	0	0	0	0	1	0

QoS level

未使用。

DUP flag

未使用

RETAIN flag

未使用

Remaining Length field

这是可变头的长度（2 字节）。它可以是一个多字节字段。

3.7.2. Variable header

Variable header包含与被确认的PUBREL消息相同的Message ID.

bit	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

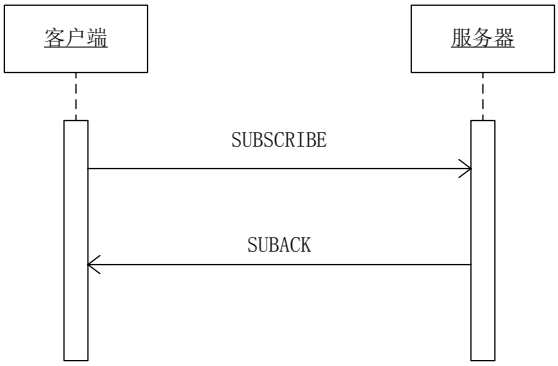
3.7.3. Payload

无 payload.

3.7.4. Actions

当客户端（应该为发布者）收到PUBCOMP消息后，就会丢弃原消息(即PUBLISH消息)，因为原消息已经被传递给了服务器，而且仅仅传递了一次。（QoS level=2）

3.8. SUBSCRIBE-订阅主题（可多个）



客户端用SUBSCRIBE消息可以向服务器注册一个或多个感兴趣的主题名。向这些主题发布的消息会被服务器以PUBLISH消息的形式传递给订阅者。SUBSCRIBE消息还指定了QoS的级别，以指导订阅者接收发布的消息。

3.8.1. Fixed header

下表显示了固定头的格式：

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (8)				DUP flag	QoS level		RETAIN
	1	0	0	0	0	0	1	x
byte 2	Remaining Length							

QoS level

SUBSCRIBE消息使用QoS级别1来确认多个订阅请求，响应的SUBACK通过匹配Message ID来识别。重试的处理方式同PUBLISH消息。

DUP flag

设置为零（0）。这意味着这个消息正在被第一次发送。更多详细情况请查看DUP。

RETAIN flag

未使用

Remaining Length field

Payload 的长度。它可以是一个多字节字段。

3.8.2. Variable header

variable header包含一个Message ID，因为SUBSCRIBE消息的QoS级别为1。更多详细情况参考Message identifiers。

下表显示了一个variable header的Message ID为10的格式的例子。

	Description	7	6	5	4	3	2	1	0
Message Identifier									
byte 1	Message ID MSB (0)	0	0	0	0	0	0	0	0
byte 2	Message ID LSB (10)	0	0	0	0	1	0	1	0

3.8.3. Payload

SUBSCRIBE消息的payload包含一个客户端希望订阅的主题名列表和客户端要接收消息的QoS级别。这些字符串都是UTF编码，QoS级别占用一个单字节的2位。主题串可能包含特定的主题通配符（代表一组主题）。这些 topic/QoS成对连续包装，如下表的payload所示。

Topic name	"a/b"
Requested QoS	1
Topic name	"c/d"
Requested QoS	2

SUBSCRIBE消息中的主题名是非压缩的，payload的例子见下表：

	Description	7	6	5	4	3	2	1	0
Topic name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	'a' (0x61)	0	1	1	0	0	0	0	1
byte 4	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 5	'b' (0x62)	0	1	1	0	0	0	1	0
Requested QoS									
byte 6	Requested QoS (1)	x	x	x	x	x	x	0	1
Topic Name									
byte 7	Length MSB (0)	0	0	0	0	0	0	0	0
byte 8	Length LSB (3)	0	0	0	0	0	0	1	1

byte 9	'c' (0x63)	0	1	1	0	0	0	1	1
byte 10	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 11	'd' (0x64)	0	1	1	0	0	1	0	0
Requested QoS									
byte 12	Requested QoS (2)	x	x	x	x	x	x	1	0

假设请求的QoS级别被授权，客户端接收的PUBLISH消息的QoS级别小于或等于这个级别，PUBLISH消息的级别取决于发布者的原始消息的QoS级别。例如，如果客户端对某个特定主题有个QoS级别为1的订阅，然后一个针对该主题的QoS级别为0的消息以QoS为0的级别传递给了客户端。一个针对此主题的QoS级别为2的PUBLISH消息降级到QoS级别为1传递给了这个客户端.

这样做的必然结果就是：订阅一个QoS级别为2的主题就等于说：“对于这个主题，我想按照消息被发布的QoS级别来收这个消息”。

这意味着由发布者负责决定消息可被传递的最大QoS级别，但订阅者能够降低QoS的级别到一个更适合它使用的QoS级别。一个QoS消息时永远不会升级。被请求的QoS字段编码紧跟在每个UTF编码的主题名后面，如下表所示：

bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	QoS level	
	x	x	x	x	x	x		

最高的6位在本协议版本中没有用到，为将来预留。

如果一个请求的QoS级别占用了QoS level的两位（也就是1　1，非法的Qos级别3），会被认为是一个非法数据包，连接会被关闭。

3.8.4. Response

当服务器从客户端接收一个SUBSCRIBE消息时，服务器会返回一个SUBACK消息。服务器可能会在客户端收到SUBACK消息之前就因为这个客户端之前的订阅开始发送PUBLISH消息。（这个客户端之前订阅过，如持久订阅，一连上来，如果有消息，则服务器就会推送，而不等这次订阅的应答）

请注意：如果服务器实现中没有给一个客户端的SUBSCRIBE请求授权，是没有方法通知该客户端的。因此，它必须用SUBACK给出一个明确的确认，且客户端将不会被告知该订阅是没有被授权的。

服务器可以选择授权一个比客户端请求级别更低的QoS级别。在服务器不能提供更高等级的QoS级别时，该情况就可能发生。例如，如果服务器不能提供一个可靠的持久性机制，它可能会选择只授予QoS级别为0的订阅。

3.9. SUBACK – 订阅确认

服务器给客户端发送一个SUBACK，以证明已经收到了SUBSCRIBE消息。

SUBACK消息包含一系列的授权QoS级别，SUBACK消息中的授权QoS级别的顺序与相应的SUBSCRIBE消息中的主题名的排列顺序是一致的。

3.9.1. Fixed header

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (9)				DUP flag	QoS level		RETAIN
	1	0	0	1	x	x	x	x
byte 2	Remaining Length							

QoS level
Not used.
DUP flag
Not used.
RETAIN flag
Not used.
Remaining Length field
The length of the payload. It can be a multibyte field.

3.9.2. Variable header

variable header中包含正在被确认的SUBSCRIBE消息的Message ID。格式见下表：

	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

3.9.3. Payload

payload包含一个授权QoS级别的矢量。每个级别都对应一个相应的SUBSCRIBE消息的主题名。SUBACK消息中的QoS级别的顺序与SUBSCRIBE消息中的主题名和请求QoS对的顺序是匹配的。使用variable header中MessageID能够将SUBACK消息和相应的SUBSCRIBE消息进行匹配。

下表显示了授权 QoS 字段在一个字节中的编码。

bit	7	6	5	4	3	2	1	0
	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	QoS level	
	x	x	x	x	x	x		

这个字节的高6位在本版本的协议中没有用到，为将来预留。
下表为一个payload的例子.

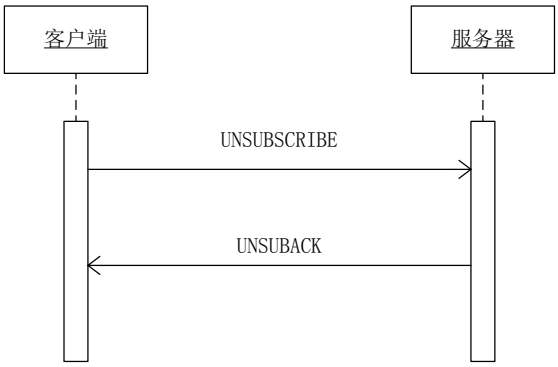
Granted QoS	0
Granted QoS	2

下表为payload的格式.

	Description	7	6	5	4	3	2	1	0
byte 1	Granted QoS (0)	x	x	x	x	x	x	0	0
byte 1	Granted QoS (2)	x	x	x	x	x	x	1	0

3.10. UNSUBSCRIBE –从主题名上取消订阅

UNSUBSCRIBE消息是由客户端向服务端发送，用于从主题名上取消订阅。



3.10.1. Fixed header

下表是 fixed header 格式的例子

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (10)				DUP flag	QoS level		RETAIN
	1	0	1	0	0	0	1	x
byte 2	Remaining Length							

QoS level

UNSUBSCRIBE消息用QoS level 1来确认多个取消订阅请求。相应的UNSUBACK消息使用这个MessageID来进行识别。

重试按照PUBLISH消息的方式进行处理。

DUP flag

设置为zero (0)．意味着这个正在发生的消息是第一次发生，详细请参见DUP

RETAIN flag

没有用到.

Remaining Length

Payload 的长度，可以是多字节的字段.

3.10.2. Variable header

variable header包含一个Message ID，因为UNSUBSCRIBE消息的QoS级别为1．具体请参见Message identifiers.

下表为一个Message ID为10的variable header的例子.

	Description	7	6	5	4	3	2	1	0
Message Identifier									
byte 1	Message ID MSB (0)	0	0	0	0	0	0	0	0
byte 2	Message ID LSB (10)	0	0	0	0	1	0	1	0

3.10.3. Payload

客户端在payload里取消对一系列主题的取消订阅。字符串都是UTF编码格式，且紧挨着一起打包。

UNSUBSCRIBE消息里的主题名是非压缩的. 下表显示了一个payload的例子.

Topic Name	"a/b"
Topic Name	"c/d"

下表显示了这个payload的格式.

	Description	7	6	5	4	3	2	1	0
Topic Name									
byte 1	Length MSB (0)	0	0	0	0	0	0	0	0
byte 2	Length LSB (3)	0	0	0	0	0	0	1	1
byte 3	'a' (0x61)	0	1	1	0	0	0	0	1
byte 4	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 5	'b' (0x62)	0	1	1	0	0	0	1	0
Topic Name									
byte 6	Length MSB (0)	0	0	0	0	0	0	0	0
byte 7	Length LSB (3)	0	0	0	0	0	0	1	1
byte 8	'c' (0x63)	0	1	1	0	0	0	1	1
byte 9	'/' (0x2F)	0	0	1	0	1	1	1	1
byte 10	'd' (0x64)	0	1	1	0	0	1	0	0

3.10.4. Response

服务器为响应UNSUBSCRIBE消息，会给客户端发送一个UNSUBACK.

3.11. UNSUBACK-取消订阅确认

服务器发送UNSUBACK消息用以确认已经收到了UNSUBSCRIBE消息.

3.11.1. Fixed header

下表是 fixed header 格式.

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (11)				DUP flag	QoS level		RETAIN
	1	0	1	1	x	x	x	x
byte 2	Remaining length (2)							
	0	0	0	0	0	0	1	0

QoS level
Not used.
DUP flag
Not used.
RETAIN flag
Not used.
Remaining Length
Variable Header的长度. (2 bytes)

3.11.2. Variable header

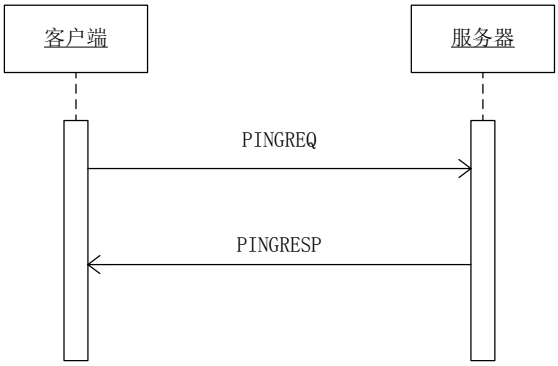
variable header包含正在被确认的UNSUBSCRIBE消息的Message ID. 下表是variable header的格式

bit	7	6	5	4	3	2	1	0
byte 1	Message ID MSB							
byte 2	Message ID LSB							

3.11.3. Payload（无）

没有 payload.

3.12. PINGREQ-PING 请求



PINGREQ消息是一个从客户端发送到服务器的“are you alive?”消息。（类似心跳包）
更多详细情况参看 Keep Alive timer。

3.12.1. Fixed header

fixed header 的格式.

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (12)				DUP flag	QoS level		RETAIN
	1	1	0	0	x	x	x	x
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

DUP, QoS和RETAIN flags都没有用到.

3.12.2. Variable header（无）

无variable header.

3.12.3. Payload（无）

无payload.

3.12.4. Response

对 PINGREQ 消息的响应是 PINGRESP 消息.

3.13. PINGRESP – PING 响应

PINGRESP消息是从服务器发送给PINGREQ消息的响应，意思是“yes ，I am alive”。

更多细节参看Keep Alive timer

3.13.1. Fixed header

下表显示了Fixed header的格式。

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (13)				DUP flag	QoS level		RETAIN
	1	1	0	1	x	x	x	x

bit	7	6	5	4	3	2	1	0
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

DUP, QoS和AIN flags都没有用到

3.13.2. Variable header（无）

无variable header.

3.13.3. Payload（无）

无payload.

3.14. DISCONNECT-断开连接通知

DISCONNECT消息是从客户端发送到服务器，以表明它即将关闭TCP/IP连接。这种考虑到了clean disconnection（干净的断开），而不仅仅是拔掉网线。

如果客户端连接时使用了clean session标志，那么这个客户端之前所维护的信息将会被丢弃。（类似非持久）

服务端收到DISCONNECT后，不应该依赖客户端来关闭TCP/IP连接。

3.14.1. Fixed header

fixed header格式见下表：

bit	7	6	5	4	3	2	1	0
byte 1	Message Type (14)				DUP flag	QoS level		RETAIN
	1	1	1	0	x	x	x	x
byte 2	Remaining Length (0)							
	0	0	0	0	0	0	0	0

DUP, QoS和RETAIN flags都没有在DISCONNECT消息中用到.

3.14.2. Variable header（无）

无variable header.

3.14.3. Payload（无）

无payload.

4. Flows 协议流

4.1. Quality of Service levels and flows

（服务等级的质量和流程）

MQTT是根据定义在QoS中的等级传递消息的。等级描述如下：

4.1.1. QoS level 0: At most once delivery（最多一次）

(类似无需应答的发送)

消息的传递完全依赖底层的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。

下表描述了QoS level 0的协议流：

Client	Message and direction	Server
QoS = 0	PUBLISH ---------->	**Action:** Publish message to subscribers

4.1.2. QoS level 1: At least once delivery（至少传递一次）

服务器的消息接收由PUBACK消息进行确认。如果通信链路异常、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DUP位的消息。

消息到达服务器至少一次。SUBSCRIBE和UNSUBSCRIBE消息使用QoS级别1。QoS级别为1的消息的消息头中都有一个Message ID。

下表显示了QoS level 1协议流。

Client	Message and direction	Server
QoS = 1 DUP = 0 Message ID = x **Action:** Store message	PUBLISH ---------->	**Actions:** • Store message • Publish message to subscribers • Delete message
Action: Discard message	PUBACK <-+--------	

如果客户端没有收到PUBACK消息（在应用中定义的时间范围内，或者检测到了一个失败且通讯会话被重启），客户端可能重发这条设置了DUP标志的PUBLISH消息。

当从客户端收到了一条重复的消息，服务器会向订阅者重新发布消息，且会发送另外一个PUBACK消息。

4.1.3. QoS level 2: Exactly once delivery（只传递一次）

在QoS level 1之上的附加协议流能够确保重复的消息不会被传递给正在接收的应用程序。当不允许出现重复消息时，这就是最高级别的传递。这会增加网络流量，但是通常来讲是可接受的，因为消息内容

很重要。

一个QoS level 2的消息在消息头中有一个Message ID。

下表显示了QoS level 2协议流。PUBLISH流被接收者处理流程有两种语义。不同的流程，消息何时可以对订阅者可见的时间点就不同。语义的选择完全由实施决定，且不会影响QoS level 2协议流的保证效果。

Client	Message and direction	Server
QoS = 2 DUP = 0 Message ID = x **Action:** Store message	PUBLISH ---------->	**Action:** Store message *or* **Actions:** • Store message ID • Publish message to subscribers
	PUBREC <----------	Message ID = x
Message ID = x	PUBREL ---------->	**Actions:** • Publish message to subscribers • Delete message *or* **Action:** Delete message ID
Action: Discard message	PUBCOMP <----------	Message ID = x

如果检测到了异常、或者超时，协议流会从最后一个没被确认的消息开始重试PUBLISH或PUBREL。详细请参见Message delivery retry。附加的协议流保证了消息只会传递给订阅者一次。

4.1.4. Assumptions for QoS levels 1 and 2

任何网络中都可能发生设备或通讯链路的失败，链路的一端可能不知道另一端发生了什么，这些被称为“可疑（in doubt）”窗口。在这些场景中，必须对消息传递相关的设备和网络的可靠性做一些假设。MQTT假设客户端和服务器通常都是可靠的，通讯通道更可能是不可靠的。如果客户端设备发生故障，将

是灾难性而不是一个暂时性的故障。从设备恢复数据的可能性非常低。有些设备有非易失性(non-volatile ）存储，如flash ROM。在客户端设备上提供更持久的存储可以保护大多数重要的数据不受某些异常的而影响。除了基本的通讯线路异常外，异常模式矩阵变得很复杂，很多场景是MQTT所无法处理的。

4.2. Message delivery retry

尽管TCP通常能够保证数据包的传递，在某些特定场景下，MQTT消息可能会收不到。可能无法接收MQTT消息。在这种场景下，MQTT消息会期望有一个响应（QoS>0 PUBLISH, PUBREL, SUBSCRIBE, UNSUBSCRIBE）,如果响应在特定时间内没有收到，发送者可以重试发送。发送者应该在消息上设置DUP标识。

重试的次数应该是一个可配置的选项。然而，应该保证消息在发送过程中不会超时，例如，在一个网速很慢的网上发送一个大消息所花费的时间自然会比在快速网络上发送一个小消息所花费的时间长。多次重试发送一个超时的消息经常会使事情变得更糟，所以，在多次重试的情况下，应当采用增加超时时间值的策略。当客户端重连，如果没有被标记为干净会话（clean session），那么客户端和服务端都应该重传任何之前正在传递（in-flight）中的消息。

除了 “重连”这种重试行为，客户并没有被要求重传消息。然而，broker 应该重试任何未确认的消息。

4.3. Message ordering

Message 顺序受到很多因素的影响，包括：客户端允许同时运行多少 in-flight PUBLISH 流，客户端的是单线程还是多线程。为了讨论的目的 ,假设客户端的某个数据包“MQTT V3.1 Protocol Specification“在向网络上写和从网络上读的时间点上是单线程的，为了保证消息的顺序，必须保证每个消息传递流的存储都是按照想要的顺序。例如，在 QoS level 2 流程中，每个 PUBREL 流的发送顺序必须完全按照最初的 PUBLISH 流：

（也就是说，每个工作流程都不能乱，都是完全按照 PUBLISH 的流程进行，虽然有的消息的某个步骤快，有的慢）

Client	Message and direction	Server
	PUBLISH 1 ---------->	
	PUBLISH 2 ---------->	
	PUBLISH 3 ---------->	
	PUBREC 1 <----------	
	PUBREC 2 <----------	
	PUBREL 1 ---------->	
	PUBREC 3 <----------	
	PUBREL 2 ---------->	
	PUBCOMP 1 <----------	
	PUBREL 3 ---------->	
	PUBCOMP 2 <----------	
	PUBCOMP 3 <----------	

允许的in-flight消息的数量对可以采用的保证措施的类型有影响：

- in-flight窗口等于1，每个传递流在下一个开始之前完成。这就保证消息是以提交的顺序进行传递。
- in-flight 窗口大于 1，消息顺序只能用在 QoS level 内进行保证。（见上图）

Appendix A – Topic wildcards

订阅可以包含特殊字符，这样就可以一下子订阅多个主题。主题层次分隔符是用来引入主题的结构。多层次的通配符和单层次的通配符可以被用于订阅，但是不能用于发布消息的主题。

Topic level separator 主题层次分隔符

反斜线（/)用来在主题树内分割每一层，为主题空间提供层次结构。当订阅者指定主题树用到了两个通配符时，主题层次分隔符就显得很重要。

Multi-level wildcard 多层通配符

是一个通配符，可以在主题内匹配任何数量的层次。比方说，如果你订阅了finance/stock/ibm/#，你会收到这些主题相关的消息：

- finance/stock/ibm
- finance/stock/ibm/closingprice
- finance/stock/ibm/currentprice

多层匹配符可以表示0或更多层，所以*finance/#*可以匹配*finance*，*#*表示0层。层次分隔符在上下文中没有意义，因为没有层可以分割。

多层匹配符必须是主题树的最后一个字符。

例如：

- *finance/#*是有效的；
- *finance/#/closingprice*是无效的.

Single-level wildcard 单层通配符

加号（+）是一个只能匹配一个主题级别的通配符。例如，finance/stock/+可以匹配finance/stock/ibm和finance/stock/xyz，而不是finance/stock/ibm/closingprice。同时，由于单层通配符只能匹配一层，则finance/+不匹配finance。

单级通配符可以在任何层次的主题树中使用，还可以与多层通配符一起使用。单层通配符使用时必须紧跟层次分隔符，除非是当它自己指定的时？？+和finance/+都有效，但finance+是无效的。

单层通配符可以用于主题树的结尾或是主题树内。例如，finance/+和finance/+/ibm都是有效的。

Topic semantics and usage 主题的语法和用途

当构建一个应用时，应该从主题名的语法和语义上考虑以下原则：

- 一个主题必须至少有一个字符长。
- 主题名称是区分大小写的。例如，ACCOUNTS和Accounts是两个不同的主题。
- 主题名称可以包含空格字符，例如Accounts payable是一个合法的主题。
- 斜杠 “/” 创造了一个截然不同的主题. 例如，/finance 与finance不同。/finance可以匹配”+/+”和”/+”，但不能匹配”+”。
- 不要在任何主题中包含空字符（Unicode x0000）。

以下原则适用于主题树的构建和内容：

- 长度限制在64K以内，但是在这个范围内，主题树内的级别数没有限制。
- 可以有任意数量的根节点，也就是说，可以有任意数量的主题树。