物联网通信协议 MQTT 学习手册

1、MQTT 协议是 IOT（Internet of Things）领域的一个主流协议

在物联网的时代，每一个传感器每一个设备都想接入互联网进行数据交换。MQTT 协议非常适合这样的场合。目前国内的主流 IOT 服务器供应商均提供对 MQTT 协议的解析比如百度云计算，阿里云计算等。MQTT 协议的实现也非常简单，对带宽的要求不高，对网络链接的可靠性要求也不高，而且协议本身制定了一定的机制来处理突发事件。

MQTT 协议不仅可以在物联网领域发挥重要作用，同时也可以用于多台机器之间的信息交换比如一个车间里面所有的传感器之间数据的交换。

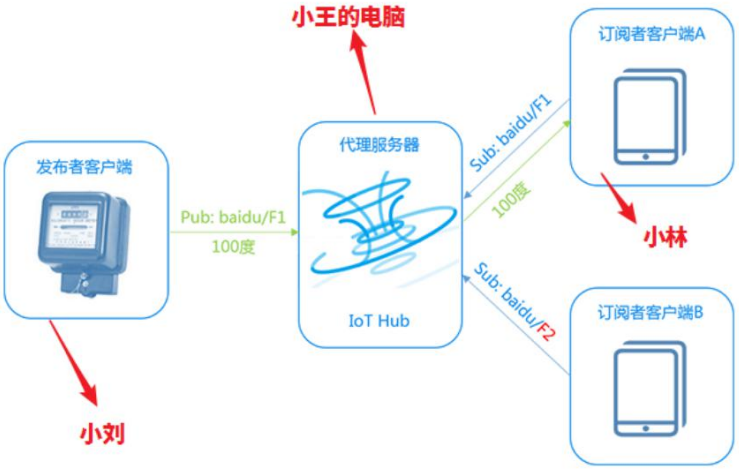
MQTT 协议也不仅仅局限于运行在互联网通信上。它是一个通信规则，对通信方式的实现不关心。通常我们提到物联网指的是通过 TCP/IP 的方式实现了通信，也就是利用互联网实现，因为互联网可以提供一个非常可靠的双向通信。

本学习手册根据 MQTT V3.1.1 版本编写
官方手册下载地址 ：http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.doc
下面章节大部分内容均参考此官方手册。

2、MQTT 通信协议需要有三个角色参与

这段文字参考百度云计算的帮助文档：
https://cloud.baidu.com/doc/IOT/MQTTProtocol.html#.E0.F6.0C.38.86.9F.BE.F8.FD.AC.D9.00.29.12.24.B6

MQTT 协议提到的一个名词 “主题”，类似于文件夹的名字一样。比如小王是电脑的主人，他的电脑上面有 a,b 两个文件夹，小刘每次存储的文件喜欢放到 a 文件夹，小宋每次存储的文件喜欢放到 b 文件夹。那么当小林想看小刘的文件时，只需要看 a 文件夹就可以了。上述的“a”文件夹的名称，在 MQTT 协议里面称作 主题 。



左图诠释了 MQTT 协议里的三个角色：发布者客户端（负责发送消息），代理服务器（负责接收和分发消息），订阅者客户端（负责接收消息）。

在 MQTT 协议里，“主题”就是一个文件夹，发布的消息可以送到一个“主题”里面，订阅者也可以从“主题”里面读取到消息。

代理服务器在国内有百度的 IoT Hub ，也有阿里云的 IoT Hub，还有很多其他品牌的服务器。

发布者客户端和订阅者客户端既可以是同一台设备，也可以是不同的设备，只要这台设备可以通过服务器的认证，并且遵循 MQTT 协议，就可以发布或者订阅消息。本学习手册的重要内容就是两个客户端如何与服务器“交流”。

3、MQTT 通信协议和大数据

(1) 小刘采集的信号是温度信息，他每间隔 1 分钟就上传一次温度信息到服务器，同时他发送的主题是 a 。
(2) 服务器接收到小刘的温度信息后，会查找当前都有哪些订阅者想看主题是 a 的信息。
(3) 小林订阅了主题是 a 的内容，只要小刘发送一次信息，小林就可以立马接收到对应的信息。
(4) 小刘和小林都需要事先通过账号密码的方式连接到服务器。小刘就像在野外工作的工作人员辛苦采集信号，而小林就像在办公室的老板千里之外洞察前线的一手信息。
(5) 如果有 1000 个小林这样的角色不停地给服务器发送温度数据。我们都知道服务器有数据保存和数据处理的能力，这时候就可以结合机器学习的相关知识去处理和分析这些数据，从而为人类的决策提供参考。

4、在 MQTT 通信协议里，字符串需要遵守 UTF-8 编码规范

在 MQTT 通信协议里，数据传送是以 Bit（位）为单位的，和我们常见的 TTL 串口类似不过他们本质上不是一个东西。MQTT 协议约定：数据传送时，高字节在前，同时，每个字节里面的最高位先传输。

UTF-8 编码规范简单点理解的话，可以看作在一个字符串的前方加了一个长度的表示。如下表所示。

字符串 “JiXiaoXin”的 UTF-8 写法

Bit	7	6	5	4	3	2	1	0
byte 1	字符串长度的高 8 位					0X00		
byte 2	字符串长度的低 8 位					0X09		
byte 3	字符串数据“J”					0X4A		
byte 4	字符串数据“i”					0X69		
byte 5	字符串数据“X”					0X58		
byte 6	字符串数据“i”					0X69		
byte 7	字符串数据“a”					0X61		
byte 8	字符串数据“o”					0X6F		
byte 9	字符串数据“X”					0X58		
byte 10	字符串数据“i”					0X69		
byte 11	字符串数据“n”					0X6E		

- 前两个字节表示一个 16Bit 的无符号型整数。这里的 0X0009 表示后续跟随了 9 个字符（这 9 个字符用 ASCII 码表示的）。
- 这个编码规范约束了字符串的最大长度为 65536.

在 MQTT 通信里，凡是涉及到一个字符串的发送或者接收，都需要用到 UTF-8 编码规范，比如 主题的名称，客户端的名称，服务器的登录账号，服务器的登录密码等。

5、MQTT 一帧消息包含的内容

MQTT 的一帧典型的消息最多由三部分组成：

固定头（所有的消息必须包含） + 可变头（有些没有） + 有效内容（有些没有）

MQTT 协议约定，根据不同的功能实现，固定头是必须要的，其他两部分内容可有可无（比如心跳包的发送和接收只要固定头即可，而从机发送的链接请求则包含了三个部分）。

5.1 固定头很重要，发送一帧消息靠它表达含义

一个固定头包含的内容

Bit	7	6	5	4	3	2	1	0
byte 1（一定有）	MQTT 控制包类型				对应左边的控制包，一些标志位的设置			
byte 2（一定有）	固定头后面所有字节的长度							
byte （可能有）	.....							
byte （可能有）	.....							
byte（可能有）	固定头后面所有字节的长度。							

- 固定头除了第一个字节决定了 MQTT 控制包的类型之外，剩下的几个字节描述了这帧消息除了固定头本身之外的所有字节的数量。它是一个特殊的编码格式，可能占一个字节，也可能占三个字节，最多占 4 个字节。

接下来分别说明一下固定头第一个字节的含义和“剩下所有字节长度”。

5.1.1 固定头的第一个字节

MQTT 控制包类型是第一个字节的 Bit7-Bit4，一共占用 4 个位，可以表示 16 种不同的组合。

MQTT 控制包类型 （一共有 14 种不同的类型）

名字	十进制	信息传输方向	说明
保留，暂时没用	0	无	保留，暂时没用
CONNECT	1	客户端 到 服务器	客户端请求连接到服务器
CONNACK	2	服务器 到 客户端	连接确认
PUBLISH	3	双向	发布消息
PUBACK	4	双向	发布消息确认
PUBREC	5	双向	Publish received (assured delivery part 1)
PUBREL	6	双向	Publish release (assured delivery part 2)
PUBCOMP	7	双向	Publish complete (assured delivery part 3)
SUBSCRIBE	8	客户端 到 服务器	客户端请求订阅
SUBACK	9	服务器 到 客户端	订阅确认
UNSUBSCRIBE	10	客户端 到 服务器	客户端请求取消订阅
UNSUBACK	11	服务器 到 客户端	取消订阅确认
PINGREQ	12	客户端 到 服务器	PING 请求
PINGRESP	13	服务器 到 客户端	PING 响应
DISCONNECT	14	客户端 到 服务器	客户端正在断开连接
保留，暂时没用	15	无	保留，暂时没用

控制包类型的标志位，是第一个字节的 Bit3-Bit0 。除了“发布消息”的控制包类型对应的标志位比较特殊外，其他的 13 种控制包类型对应的标志位是固定的，不可变更。

如果一个设备接收到了一帧信息发现这个标志位和约定的不一致，那么必须马上断开和对方的网络链接。

14 种 MQTT 控制包类型对应的标志位

控制包类型	后面的标志位	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	固定的	0	0	0	0
CONNACK	固定的	0	0	0	0
PUBLISH	Used in MQTT 3.1.1	DUP[1]	QoS[2]	QoS[2]	RETAIN[3]
PUBACK	固定的	0	0	0	0
PUBREC	固定的	0	0	0	0
PUBREL	固定的	0	0	1	0
PUBCOMP	固定的	0	0	0	0
SUBSCRIBE	固定的	0	0	1	0
SUBACK	固定的	0	0	0	0
UNSUBSCRIBE	固定的	0	0	1	0
UNSUBACK	固定的	0	0	0	0
PINGREQ	固定的	0	0	0	0
PINGRESP	固定的	0	0	0	0
DISCONNECT	固定的	0	0	0	0

上表中的 DUP ,QoS,RETAIN 是在 MQTT V3.1.1 协议里用的，用于描述发布的这一帧消息的属性，在后文里会有详细的说明。

依据上述规则，我们可以很容易得到 14 种固定头的第一个字节的内容：（16 进制表示）

14 种固定头的第一字节内容

固定头名字	第一字节	信息传输方向	固定头含义
CONNECT	0X10	客户端 到 服务器	客户端请求连接到服务器
CONNACK	0X20	服务器 到 客户端	连接确认
PUBLISH	0X30	双向	发布消息（依据发布消息不同而不同）
PUBACK	0X40	双向	发布消息确认
PUBREC	0X50	双向	Publish received (assured delivery part 1)
PUBREL	0X62	双向	Publish release (assured delivery part 2)
PUBCOMP	0X70	双向	Publish complete (assured delivery part 3)
SUBSCRIBE	0X82	客户端 到 服务器	客户端请求订阅
SUBACK	0X90	服务器 到 客户端	订阅确认
UNSUBSCRIBE	0XA2	客户端 到 服务器	客户端请求取消订阅
UNSUBACK	0XB0	服务器 到 客户端	取消订阅确认
PINGREQ	0XC0	客户端 到 服务器	PING 请求
PINGRESP	0XD0	服务器 到 客户端	PING 响应
DISCONNECT	0XE0	客户端 到 服务器	客户端正在断开连接

- 在最简单的应用中，可以发布一条质量等级为 0，不需要确认的消息，此时设置上表中橙色部分为 0X30.
- 后面的所讲述的内容，都和固定头的名字有关系，不同的固定头后面跟随的内容不一样。

5.1.2 “剩余长度”，描述一帧消息占用几个字节

“剩余长度”指的是 **可变头（有些没有）** + **有效内容（有些没有）** 所有字节的数量。

“剩余长度”采用的是一种 “可变长度”编码规则 ，这种编码规则可以使用较少的字节表达较多的内容。它约定，一个字节的 Bit7 位不代表具体的数字，而表示一个标志位，所以一个字节本身能表达的数值范围是：0-127 ，如果一个数字超过了 127，那么必须再用一个字节才可以表示，同时，这个字节的 Bit7 要置 1 。这个编码规则本身最多占用四个字节。举例如下：

假如数字 68 ，16 进制表示 0X44 ，大小小于 127 ，所以 编码规则和常规一样。就是 0X44.

假如数字 321 大于 127 ，所以编码需要遵守编码规则，应该是 0XC1 0X02，具体的解释如下：

Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	1	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
	0XC1								**0X02**							
	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0
	0X41(十进制表示 65)								0X02，十进制表示 2							
	第一个字节								第二个字节							
	65 + 2*128 = 321															

“可变长度”编码规则的范围

编码字节数量	最小值	最大值
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)

5.2 可变头 的简单描述

可变头标识占两个字节。一个可变头包含很多内容，可变头标识只是其中的一部分。

先简单说一下可变头标识的出现场合，后面有详细说明。可变头标识是否有必要和固定头名字有关。

固定头名字和第一字节		固定头含义	可变头标识
CONNECT	0X10	客户端请求连接到服务器	NO
CONNACK	0X20	连接确认	NO
PUBLISH	0X30	发布消息（依据发布消息不同而不同）	YES (If QoS > 0)
PUBACK	0X40	发布消息确认	YES
PUBREC	0X50	Publish received (assured delivery part 1)	YES
PUBREL	0X62	Publish release (assured delivery part 2)	YES
PUBCOMP	0X70	Publish complete (assured delivery part 3)	YES
SUBSCRIBE	0X82	客户端请求订阅	YES
SUBACK	0X90	订阅确认	YES
UNSUBSCRIBE	0XA2	客户端请求取消订阅	YES
UNSUBACK	0XB0	取消订阅确认	YES
PINGREQ	0XC0	PING 请求	NO
PINGRESP	0XD0	PING 响应	NO
DISCONNECT	0XE0	客户端正在断开连接	NO

可变头的内容里，有一个字节叫做“连接的标识符”，它描述了当前的这个连接的一些设置的内容。这个字节在客户端连接服务器时候用到的。具体如下所示：

“连接的标识符”一个字节

Bit	7	6	5	4	3	2	1	0
每一位的含义	User Name Flag	Password Flag	Will Retain	Will QoS		Will Flag	Clean Session	Reserved
取值	X	X	X	X	X	X	X	0
描述	0 无用户名 1 有用户名	0 没有密码 1 有密码	看详细解释	看详细解释		看详细解释	是否存储会话状态	必须为 0

Clean Session：如果该位被设置为 0，则该连接被认为是持久连接，其具体表现为：当该客户断开后，任何订阅的主题和 QoS 被设置为 1 或 2 的信息都会保存，直到该客户端再次连接上 server 端（百度云物接入服务支持将该消息保留 24 小时）。若“clean session”被设置为 1，当该客户断开后，所有的订阅主题都会被移除。

Will Flag：当一个客户端断开连接的时候，它希望客户端可以发送它指定的消息。该消息和普通消息的结构相同。（通过在帧消息的有效报文中设置 Will Topic 和 Will Message 实现）

Will QoS：服务器在发生意外的情况下发送遗嘱消息的服务等级。如果 Will Flag 是 0，那么这个必须是 0。

Will Retain：遗嘱保留，如果勾选遗嘱保留，遗嘱消息发布时将会保留且发送给新的订阅消息。

什么是临终遗嘱？

MQTT 协议利用 KeepAlive 机制在客户端异常断开时发现问题。当客户端断开时（例如：电量耗尽、系统崩溃或者网络断开），代理服务器会采取相应措施。客户端设置“临终遗嘱”（LWT）信息后，当代理服务器检测到客户端离线后，就会发送保存在特定主题上的 LWT 信息，让其它订阅该主题的客户端知道该节点已经意外离线。

发布者客户端通过设置 PUBLISH 报文中的 QoS 标志位，对于客户端发布的消息提供三种服务质量等级，如下：
QoS=0，协议对此等级应用信息不要求回应确认，也没有重发机制，这类信息可能会发生消息丢失或重复，取决于 TCP/IP 提供的尽最大努力交互的数据包服务。
QoS=1，确保信息到达，但消息重复可能发生，发送者如果在指定时间内没有收到 PUBACK 控制报文，应用信息会被重新发送。
QoS=2，最高级别的服务质量，消息丢失和重复都是不可接受的。

5.2 帧内容 的简单描述

帧内容 依据传输内容的不一样，所占字节的长度也不一样。比如会包含用户名，密码的信息，不同的服务器不同的用户肯定不一样。注意：帧内容里的数据主要是字符串，需要符合 UTF-8 编码规范。

简单说一下帧内容的出现场合

固定头名字和第一字节		固定头含义	可变头标识	帧内容
CONNECT	0X10	客户端请求连接到服务器	NO	必须要有
CONNACK	0X20	连接确认	NO	None
PUBLISH	0X30	发布消息（依据发布消息不同而不同）	YES (If QoS > 0)	看情况变化
PUBACK	0X40	发布消息确认	YES	None
PUBREC	0X50	Publish received (assured delivery part 1)	YES	None
PUBREL	0X62	Publish release (assured delivery part 2)	YES	None
PUBCOMP	0X70	Publish complete (assured delivery part 3)	YES	None
SUBSCRIBE	0X82	客户端请求订阅	YES	必须要有
SUBACK	0X90	订阅确认	YES	必须要有
UNSUBSCRIBE	0XA2	客户端请求取消订阅	YES	必须要有
UNSUBACK	0XB0	取消订阅确认	YES	None
PINGREQ	0XC0	PING 请求	NO	None
PINGRESP	0XD0	PING 响应	NO	None
DISCONNECT	0XE0	客户端正在断开连接	NO	None

6、MQTT 连接服务器 + 心跳包

6.1 “CONNECT”，客户端发送给服务器的连接请求。

当确保网络连通后，客户端首先需要连接到服务器。如果连接成功，服务器需要有一个链接成功的返回。如果连接成功，客户端就不需要再发送链接请求了，只需要发送数据到服务器即可，同时发送必要的心跳包来保持连接。

验证网络是否连通的方法很简单，只需要发送“Ping 请求”到服务器，如果服务器有响应就证明网络链接是可靠的。发送（16 进制格式） C0 00 ，服务器必须返回（16 进制格式） D0 00 。

“**固定头**”： 第一个字节肯定是 0X10 ，后续的**帧长度**要看后面跟随多少信息，待定（0X53）。

“**可变头**”：协议名（UTF-8 编码）+协议版本 1 字节+连接的标识符 1 字节+心跳包时间 2 字节

一个完整的**可变头**如下表所示

字节	**描述**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
协议名称（固定值）									
byte 1 (0X00)	(0X00)	0	0	0	0	0	0	0	0
byte 2 (0X04)	Length LSB (0X04)	0	0	0	0	0	1	0	0
byte 3 (0X4D)	‘M’	0	1	0	0	1	1	0	1
byte 4 (0X51)	‘Q’	0	1	0	1	0	0	0	1
byte 5 (0X54)	‘T’	0	1	0	1	0	1	0	0
byte 6 (0X54)	‘T’	0	1	0	1	0	1	0	0
协议版本（固定值）									
	Description	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 7 (0X04)	Level (4)	0	0	0	0	0	1	0	0
连接的标识符（有用户名，有密码，客户端掉线后服务器清空客户端的信息）									
byte 8 (0XC2)	User Name Flag (1) Password Flag (1) Will Retain (0) Will QoS (00) Will Flag (0) Clean Session (1) *Reserved* (0)	1	1	0	0	0	0	1	0
心跳包时间设置，单位是 S (这里表示 60 秒，意思是如果客户端超过 60S 没有发送有效信息到服务器，那么服务器认为这个客户端已失联，会主动断开与客户端的连接)									
byte 9 (0X00)	Keep Alive MSB (0)	0	0	0	0	0	0	0	0
byte 10 (0X3C)	Keep Alive LSB (10)	0	0	1	1	1	1	0	0

“**有效内容**”： “用户 ID” + “临终消息主题” + “临终消息” + “用户名” + “密码”

- 用户 ID 必须保持唯一，在一个服务器上的所有设备，每个设备的 ID 都不一样；

- 应该从服务器那里获取用户名和密码；
- “临终消息主题”和“临终消息”如果可变头里面的 连接标识符没有允许，就不要添加了。

注意：我们在测试之前需要配置好服务器，然后获取一个用户名和密码。比如下面这样的格式：
用户名：jixin/jixiaoxin 用户 ID：Ling_Yao(用户 ID 是自己起的名字)
密码：ymjohJfqMO9KFzjKhVqeR78wnRpt0U0Xxrqq5VEHdcI=

依据上述的设置，本例子在连接服务器的时候，有效内容应该如下表所示:

字节		描述		发送密码，UTF8 编码		
发送用户 ID，UTF8 编码				Byte36	0X66	f
Byte1	0X00	Length MSB		Byte37	0X71	q
Byte2	0X08	Length LSB		Byte38	0X4D	M
Byte3	0X4C	L		Byte39	0X4F	0
Byte4	0X69	i		Byte40	0X39	9
Byte5	0X6E	n		Byte41	0X4B	K
Byte6	0X67	g		Byte42	0X46	F
Byte7	0X5F	_		Byte43	0X7A	z
Byte8	0X59	Y		Byte44	0X6A	j
Byte9	0X61	a		Byte45	0X4B	K
Byte10	0X6F	o		Byte46	0X68	h
				Byte47	0X56	V
发送用户名，UTF8 编码				Byte48	0X71	q
Byte11	0X00	Length MSB		Byte49	0X65	e
Byte12	0X0F	Length LSB		Byte50	0X52	R
Byte13	0X6A	j		Byte51	0X37	7
Byte14	0X69	i		Byte52	0X38	8
Byte15	0X78	x		Byte53	0X 77	w
Byte16	0X69	i		Byte54	0X6E	n
Byte17	0X6E	n		Byte55	0X52	R
Byte18	0X2F	/		Byte56	0X70	p
Byte19	0X6A	j		Byte57	0X74	t
Byte20	0X69	i		Byte58	0X30	0
Byte21	0X78	x		Byte59	0X55	U
Byte22	0X69	i		Byte60	0X30	0
Byte23	0X61	a		Byte61	0X58	X
Byte24	0X6F	o		Byte62	0X78	x
Byte25	0X78	x		Byte63	0X72	r
Byte26	0X69	i		Byte64	0X71	q
Byte27	0X6E	n		Byte65	0X71	q
				Byte66	0X35	5
发送密码，UTF8 编码				Byte67	0X56	V
Byte28	0X00	Length MSB		Byte68	0X45	E
Byte29	0X2C	Length LSB		Byte69	0X48	H

Byte30	0X79	y		Byte70	0X64	d
Byte31	0X6D	m		Byte71	0X63	c
Byte32	0X6A	j		Byte72	0X49	I
Byte33	0X6F	o		Byte73	0X3D	=
Byte34	0X68	n				
Byte35	0X4A	J				

至此，连接服务器需要发送的信息都已经完成了。那么我们计算一下一共需要发送多少个字节，然后就可以确定固定头的 “剩余字节数” 的参数了。

可变头 10 个字节+有效信息 73 个字节，一共是 83 个字节，所以 “剩余字节数应该是” 0X53

所以，发送以下信息到服务器即可：（16 进制格式）

10 53

00 04 4D 51 54 54 04 C2 00 3C

00 08 4C 69 6E 67 5F 59 61 6F

00 0F 6A 69 78 69 6E 2F 6A 69 78 69 61 6F 78 69 6E

00 2C 79 6D 6A 6F 68 4A 66 71 4D 4F 39 4B 46 7A 6A 4B 68 56 71 65 52 37 38 77 6E 52 70 74 30 55 30 58 78 72 71 71 35 56 45 48 64 63 49 3D

发送成功之后，服务器会返回（16 进制格式） 20 02 00 00

如果接收到上述返回的消息，证明已经成功和服务器建立了可靠连接。分析一下来自服务器的消息含义：第一个字节 20 可以从固定头那一节查到，是服务器发送的数据，叫做“连接确认”；第二个字节 02 表示后面还有两个数据；后面的两个数据 00 00 ，表示两个有效数据。后面发送两个数据的一层意思是，验证从机连接的协议是否正确，即从机接收到 02 这个数据后应该判断是否真的接收到了两个数据，如果不是，那证明通信时有问题的。

6.2 按时发送心跳包，和服务器保持联系

其实这里的心跳包，是通过发送 “Ping 请求” 给服务器来实现的。在连接服务器的时候，我们设置了心跳包时间为 60S，那么我们需要每 60S 之内就和服务器“Ping 请求”一次，证明网络链接是可靠的，如果接收不到服务器的返回，那么可能是我们的网络掉线了。

发送： C0 00 接收： D0 00