XRADIO MQTT Developer Guide

Revision 1.0

Otc 23, 2019

Declaration

THIS DOCUMENTATION IS THE ORIGINAL WORK AND COPYRIGHTED PROPERTY OF XRADIO TECHNOLOGY (“XRADIO”). REPRODUCTION IN WHOLE OR IN PART MUST OBTAIN THE WRITTEN APPROVAL OF XRADIO AND GIVE CLEAR ACKNOWLEDGEMENT TO THE COPYRIGHT OWNER.

THE PURCHASED PRODUCTS, SERVICES AND FEATURES ARE STIPULATED BY THE CONTRACT MADE BETWEEN XRADIO AND THE CUSTOMER. PLEASE READ THE TERMS AND CONDITIONS OF THE CONTRACT AND RELEVANT INSTRUCTIONS CAREFULLY BEFORE USING, AND FOLLOW THE INSTRUCTIONS IN THIS DOCUMENTATION STRICTLY. XRADIO ASSUMES NO RESPONSIBILITY FOR THE CONSEQUENCES OF IMPROPER USE (INCLUDING BUT NOT LIMITED TO OVERVOLTAGE, OVERCLOCK, OR EXCESSIVE TEMPERATURE).

THE INFORMATION FURNISHED BY XRADIO IS PROVIDED JUST AS A REFERENCE OR TYPICAL APPLICATIONS, ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS DOCUMENT DO NOT CONSTITUTE A WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. XRADIO RESERVES THE RIGHT TO MAKE CHANGES IN CIRCUIT DESIGN AND/OR SPECIFICATIONS AT ANY TIME WITHOUT NOTICE.

NOR FOR ANY INFRINGEMENTS OF PATENTS OR OTHER RIGHTS OF THE THIRD PARTIES WHICH MAY RESULT FROM ITS USE. NO LICENSE IS GRANTED BY IMPLICATION OR OTHERWISE UNDER ANY PATENT OR PATENT RIGHTS OF XRADIO.THIRD PARTY LICENCES MAY BE REQUIRED TO IMPLEMENT THE SOLUTION/PRODUCT. CUSTOMERS SHALL BE SOLELY RESPONSIBLE TO OBTAIN ALL APPROPRIATELY REQUIRED THIRD PARTY LICENCES. XRADIO SHALL NOT BE LIABLE FOR ANY LICENCE FEE OR ROYALTY DUE IN RESPECT OF ANY REQUIRED THIRD PARTY LICENCE. XRADIO SHALL HAVE NO WARRANTY, INDEMNITY OR OTHER OBLIGATIONS WITH RESPECT TO MATTERS COVERED UNDER ANY REQUIRED THIRD PARTY LICENCE.

Revision History

Version	Date	Summary of Changes
1.0	2019-10-24	Initial Version

Contents

1 模块概要

1.1 功能介绍

MQTT 是一个客户端服务端架构的发布/订阅模式的消息传输协议。它的设计思想是轻巧、开放、简单、规范，因此易于实现。这些特点使得它对很多场景来说都是很好的选择，包括受限的环境如机器与机器的通信（M2M）以及物联网环境（IoT），这些场景要求很小的代码封装或者网络带宽非常昂贵。SDK 中的 mqtt 基于开源代码“eclipse paho.mqtt.embedded-c v1.0.0”，并修复若干 bug。

1.2 代码位置

模块	文件类型	文件位置
MQTT	source	sdk/src/net/mqtt/
	header	sdk/include/net/mqtt/
	demo	sdk/project/common/cmd/cmd_mqtt.c sdk/project/demo/aliyun_demo sdk/project/example/mqtt

2 模块结构体

2.1 客户端结构体

该结构体表示一个客户端实例，由于 mqtt 开源代码的设计缺陷，该结构体大部分是可以用户自己设置的，少部分是 mqtt 模块内部使用，用户无需设置的。

```
struct Client {
    unsigned int next_packetid; //mqtt 模块内部使用，用户无需设置
    unsigned int command_timeout_ms; //命令传输超时时间，用户可设置
    size_t buf_size, readbuf_size; //发送缓冲区和接收缓冲区大小，用户必须设置
    unsigned char *buf; //发送缓冲区指针，需要外部传入，用户必须设置
    unsigned char *readbuf; //接收缓冲区，需要外部传入，用户必须设置
    unsigned int keepAliveInterval; // 心跳时间，用户无需设置，需在
MQTTPacket_connectData 结构体中设置
    char ping_outstanding; //ping 是否发送标志位，mqtt 模块内部使用，用户无需设置
    int isconnected; //mqtt 是否连接成功标志位，mqtt 模块内部使用，用户无需设置
    struct MessageHandlers
    {
        const char* topicFilter;
        void (*fp) (MessageData*);
    } messageHandlers[MAX_MESSAGE_HANDLERS]; // 存储订阅的 topic，mqtt 模块内部使用，
用户无需设置
    void (*defaultMessageHandler) (MessageData*); //消息发布失败后，调用的回调函数，
用户可设置
    Network* ipstack; //Network 指针，mqtt 模块内部使用，用户无需设置
    Timer last_sent, last_received; //最后发送和最后接收消息的时间，mqtt 模块内部使用，
用户无需设置
#ifdef PACKET_SPLICE_BUGFIX
    int remain_pktfrag_len; //剩余未读取数据长度，mqtt 模块内部使用，用户无需设置
#endif
};
```

2.2 连接包配置结构体

该结构体是连接 mqtt 时，在连接包中需要设置的参数。

```
typedef struct
{
    char struct_id[4]; //必须设置为 MQTC
    int struct_version; //结构体的版本号，必须为 0
    unsigned char MQTTVersion; //mqtt 的版本，3 = 3.1 4 = 3.1.1
    MQTTString clientID; //客户端的 id 号
    unsigned short keepAliveInterval; //心跳时间，用户必须设置
    unsigned char cleansession; //是否使能服务器的 CleanSession 选项，0：不使能，1：使
能
    unsigned char willFlag; //遗愿设置，0：无遗愿，1：有遗愿
    MQTTPacket_willOptions will; //遗愿消息
```

```
    MQTTString username; //连接时的用户名
    MQTTString password; //连接时的密码
} MQTTPacket_connectData;
```

3 模块接口

3.1 初始化网络接口

void NewNetwork(Network* n)		备注
功能	初始化网络接口	
参数	Network：网络接口结构体	
返回值	void	

3.2 初始化客户端

void MQTTClient(Client* c, Network* network, unsigned int command_timeout_ms, unsigned char* buf, size_t buf_size, unsigned char* readbuf, size_t readbuf_size)		备注
功能	初始化客户端	
参数	c：客户端实体指针，实体需由用户创建 network：网络实体指针，实体需由用户创建 command_timeout_ms：命令超时时间 buf：写缓存指针，缓存需由用户创建 buf_size：写缓存大小 readbuf：读缓存指针，缓存需由用户创建 readbuf_size：读缓存大小	
返回值	void	

3.3 连接服务器

以下接口仅用于和服务器建立 tcp 层的链接。先要进行 tcp 层连接，然后再进行 mqtt 层连接。

int ConnectNetwork(Network* n, char* addr, int port)		备注
功能	以非 tls 方式连接服务器	
参数	Network：网络接口结构体 addr：服务器地址	

	port：服务器端口号	
返回值	void	

int TLSConnectNetwork(Network *n, const char *addr, const char *port, const char *ca_crt, size_t ca_crt_len, const char *client_crt, size_t client_crt_len, const char *client_key, size_t client_key_len, const char *client_pwd, size_t client_pwd_len)		备注
功能	以 tls 方式连接服务器	
参数	Network：网络接口结构体 addr：服务器地址 port：服务器端口号 ca_crt：ca 证书 ca_crt_len：ca 证书长度 client_crt：客户端证书 client_crt_len：客户端证书长度 client_key：客户端私钥 client_key_len：客户端私钥长度 client_pwd：客户端密码 client_pwd_len：客户端密码长度	
返回值	0：成功，其他：失败	

3.4 发送连接包

int MQTTConnect (Client*c, MQTTPacket_connectData* connectData)		备注
功能	向服务器发送连接包，即 mqtt 层的连接	
参数	c：客户端实体指针 connectData：客户端的连接包配置	
返回值	0：成功，其他：失败	

3.5 发布消息

int MQTTPublish (Client* c, const char* topicName, MQTTMessage* message)	备注

功能	向服务器发布消息	
参数	c：客户端实体指针，实体需由用户创建并已初始化 topicName：发布消息的主题 message：消息	
返回值	0：成功，其他：失败	

3.6 订阅主题

int MQTTSubscribe (Client* c, const char* topicFilter, enum QoS qos, messageHandler messageHandler)		备注
功能	向服务器订阅主题	
参数	c：客户端实体指针，实体需由用户创建并已初始化 topicFilter：订阅的主题 qos：订阅的服务质量等级 messageHandler：收到该主题消息的后，触发的回调函数	
返回值	0：成功，其他：失败	

3.7 取消订阅主题

int MQTTUnsubscribe (Client* c, const char* topicFilter)		备注
功能	取消订阅主题	
参数	c：客户端实体指针，实体需由用户创建并已初始化 topicFilter：需要取消的已订阅主题	
返回值	0：成功，其他：失败	

3.8 发送断开连接包

int MQTTDisconnect (Client* c)		备注
功能	向服务器发送断开连接的包	该接口只是向服务器发送断开连接的数据包，是 mqtt 层面的，此时 tcp 还是连接的，所以需要调用 Network 中的 disconnect 接口

参数	c：客户端实体指针，实体需由用户创建并已初始化	
返回值	0：成功，其他：失败	

3.9 Yield 接口

int MQTTYield (Client* c, int timeout_ms)		备注
功能	客户端运行，含 MQTT 帧接收、处理、发送心跳包等	该接口需要不断被调用，且该函数是阻塞式的，在 timeout 前，会阻塞同优先级的其他线程，导致其他线程不能正常运行，可根据要求自行调整 mqtt 线程优先级、减小 timeout 时间、提高调用 MQTTYield 的频率（可用系统延时），保证在每次调用的时间间隔内有时间执行其他操作，从而做到尽量减少对线程的阻塞。
参数	c：客户端实体指针，实体需由用户创建并已初始化 timeout_ms：接收阻塞时间	
返回值	0：成功，其他：失败	

4 模块例程

4.1 初始化

初始化网络实体、客户端实体。例如：命令超时时间为 30s，写缓存和读缓存设为 80Bytes，如下述代码配置。

```
Client client;
Network network;
unsigned char sendbuf[80], readbuf[80];

NewNetwork(&network);
MQTTClient(&client, &network, 30000, sendbuf, sizeof(sendbuf), readbuf,
sizeof(readbuf));
```

4.2 连接服务器

连接服务器。例如：连接 iot.eclipse.org 服务器，端口号 1883。

```
char* address = "iot.eclipse.org";
rc = ConnectNetwork(&network, address, 1883);
if (rc != 0) {
    printf("Return code from network connect is %d\n", rc);
    return -1;
}
```

或者

```
char* address = "iot.eclipse.org";
rc = TLSConnectNetwork(&network, address , 1883, NULL, 0, NULL, 0, NULL, 0, NULL,
0);
if (rc != 0) {
    printf("Return code from network connect is %d\n", rc);
    return -1;
}
```

4.3 发送连接包

连接 MQTT 服务器。例如：客户端采用 3.1.1 版本；客户端 ID 取名 XR_Radio_test0；无遗嘱；无用户名，无密码；选择清除会话。

```
MQTTPacket_connectData connectData = MQTTPacket_connectData_initializer;

connectData.MQTTVersion = 4;
connectData.clientID.cstring = "XR_Radio_test0";

if ((rc = MQTTConnect(&client, &connectData)) != 0)
    printf("Return code from MQTT connect is %d\n", rc);
```

4.4 订阅主题（可选）

订阅主题。要注意两点：

（1）首先要写一个回调函数，用于处理在该主题中收到的消息。

（2）在订阅了主题之后，主题名字需由用户一直保存，MQTT 只保存指针。此处主题名为"XR_Radio/test/#"，订阅服务质量为 QOS1。

```
void messageArrived(MessageData* data)
{
    printf("Message arrived on topic %.*s: %.*s\n",
data->topicName->lenstring.len, data->topicName->lenstring.data,
data->message->payloadlen, (char *)data->message->payload);
}
```

```
if ((rc = MQTTSubscribe(&client, "XR_Radio/test/#", QOS1, messageArrived)) != 0)
    printf("Return code from MQTT subscribe is %d\n", rc);
```

4.5 发布消息（可选）

发布消息。例如：要发布主题为"XR_Radio/test/a"，发布内容为"message number 0"，服务质量等级为 QOS1，不需要保持。

```
MQTTMessage message;
char payload[30] = "message number 0";

message.qos = QOS1;
message.retained = 0;
message.payload = payload;
message.payloadlen = strlen(payload) + 1;

if ((rc = MQTTPublish(&client, "XR_Radio/test/a", &message)) != 0)
    printf("Return code from MQTT publish is %d\n", rc);
```

4.6 取消订阅（可选）

取消订阅。例如：取消订阅"XR_Radio/test/#"。

```
if ((rc = MQTTUnsubscribe(&client, "XR_Radio/test/#")) != 0)
    printf("Return code from MQTT unsubscribe is %d\n", rc);
```

4.7 断开连接

断开 MQTT 连接与网络连接。

```
if (client.isconnected) {
    if ((rc = MQTTDisconnect(&client)) != 0) {
        printf("Return code from MQTT disconnect is %d\n", rc);
```

```
        network.disconnect(&network);
        return -1;
    }
    network.disconnect(&network);
}
```

5 注意事项

1. mqtt 模块所有的内存使用都需要外部申请，并通过指针的形式传递到 mqtt 模块，在 mqtt 反初始化时，mqtt 模块不会释放内存，需要自行释放，否则容易造成内存泄漏。

2. MQTTYield 函数是阻塞式的，在 timeout 前，会阻塞同优先级的其他线程，导致其他线程不能正常运行，可根据要求自行调整 mqtt 线程优先级、减小 timeout 时间、提高调用 MQTTYield 的频率（可用系统延时），保证在每次调用的时间间隔内有时间执行其他操作，从而做到尽量减少对线程的阻塞。

6 附录

MQTT 官方网站：http://mqtt.org/

开源代码官方网站：http://www.eclipse.org/paho/clients/c/embedded/