

The logo for RT-Thread, featuring a blue stylized wave icon above the text "RT-Thread" in a bold, blue, sans-serif font. The logo is centered within a white, rounded rectangular shape that has a slight 3D effect with a shadow. This shape is surrounded by several smaller circles in white and orange, scattered across the blue background.

RT-Thread

网络编程提升篇

MQTT : 物联网领域的即时通讯

目录

- 基础知识
- 具体示例
- 示例代码讲解
- 注意事项



基础知识

基础知识

- 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 在1999年发布。
- MQTT 协议运行在 TCP/IP 或其他网络协议之上，它将建立客户端到服务器的连接，提供两者之间的一个有序的、无损的、基于字节流的双向传输。
- MQTT 最大优点在于，可以以极少的代码和有限的带宽，为连接远程设备提供实时可靠的消息服务。作为一种低开销、低带宽占用的即时通讯协议，使其在物联网、小型设备、移动应用等方面有较广泛的应用。

基础知识

- MQTT 是一个基于客户端-服务器的消息发布/订阅传输协议。MQTT 协议是轻量、简单、开放和易于实现的，这些特点使它的适用范围非常广泛。
- MQTT可以应用在很多情况下，包括受限的环境中，如：**机器与机器通信**（M2M）和**物联网**（IoT），其中在通过卫星链路通信传感器、偶尔拨号的医疗设备、智能家居、及一些小型化设备中已广泛使用。
- 本教程就是介绍如何利用 RT-Thread 开发的 Paho MQTT 软件包与 MQTT 服务器进行通信的。

MQTT 功能介绍

- 当应用数据通过 MQTT 网络发送时，MQTT 会把与之相关的服务质量（QoS）和主题名（Topic）相关连，其特点包括：
 - 使用发布/订阅消息模式，它提供了一对多消息分发，以实现与应用程序的解耦。
 - 对负载内容屏蔽的消息传输机制。
 - 传输消息有三种服务质量（QoS）：
 - 最多一次，这一级别会发生消息丢失或重复，消息发布依赖于底层 TCP/IP 网络。即： ≤ 1
 - 至少一次，这一级别会确保消息到达，但消息可能会重复。即： ≥ 1
 - 只有一次，确保消息只有一次到达。即： $= 1$ 。在一些要求比较严格的计费系统中，可以使用此级别。
 - 数据传输和协议交换的最小化（协议头部只有2字节），以减少网络流量。
 - 通知机制，异常中断时通知传输双方。

MQTT 功能介绍

MQTT 协议中的方法

- MQTT协议中定义了一些方法（也被称为动作），用来表示对确定资源所进行操作。这个资源可以代表预先存在的数据或动态生成数据，这取决于服务器的实现。通常来说，资源指服务器上的文件或输出。
 - Connect: 等待与服务器建立连接。
 - Disconnect: 等待 MQTT 客户端完成所做的工作，并与服务器断开 TCP/IP 会话。
 - Subscribe: 等待完成订阅。
 - UnSubscribe: 等待服务器取消客户端的一个或多个 Topics 订阅。
 - Publish: MQTT 客户端发送消息请求，发送完成后返回应用程序线程。



具体示例

工程配置

- 开启 Paho MQTT 软件包
- 打开 env 工具输入 menuconfig 按照下面的路径开启 Paho MQTT 软件包

```
RT-Thread online packages
IoT - internet of things --->
[*] Paho MQTT: Eclipse Paho MQTT C/C++ client for Embedded platforms --->
```

- 进入 Paho MQTT 软件包的配置菜单按下图所示配置

```
--- Paho MQTT: Eclipse Paho MQTT C/C++ client for Embedded platforms
    MQTT mode (Pipe mode: high performance and depends on DFS) --->
[*] Enable MQTT example
[ ] Enable MQTT test
[ ] Enable support tls protocol
(4096) Set MQTT thread stack size
(1) Max pahomqtt subscribe topic handlers
[*] Enable debug log output
version (latest) --->
```

工程配置

- 配置项介绍
 - Enable MQTT example: 开启 MQTT 示例
 - Enable MQTT test: 开启测试例程
 - Enable support tls protocol: 开启 TLS 安全传输选项
 - Set MQTT thread stack size: 配置 MQTT 线程堆栈大小
 - Max pahomqtt subscribe topic handlers: 配置 MQTT 能订阅的最大 topic 主题数量
 - Enable debug log output: 开启调试日志
 - latest_version: 配置包版本选为最新版

运行示例代码

- 保存并更新软件包 `pkgs --update`
- 编译工程 `scons`
- 然后运行 `qemu`
- 系统运行起来后，在 `msh` 命令行下输入 `mq_start` 命令即可让示例代码运行。

```
msh> mq_start
```

- 运行结果如下图所示

```
[AT] ESP8266 WIFI is connected.  
[AT] AT network initialize success!  
msh />mq_start  
MQTTinter mqtt_connect_callback!  
MQTTipv4 address port: 1883  
MQTTHOST = 'iot.eclipse.org'  
msh />MQTTMQTT server connect success  
MQTTSubscribe #0 /mqtt/test OK!  
MQTTinter mqtt_online_callback!
```

预期结果

- 发布消息用命令 `mq_pub` ,用于向固定的 MQTT Topic 发送数据,同时 MQTT 服务器会立刻向该 Topic 发送同样数据, MQTT 示例测试完成,如下图所示:

```
MQTTMQTT server connect success
MQTTSubscribe #0 /mqtt/test OK!
MQTTinter mqtt online callback!
msh />mq_pub hello-rtthread
msh />MQTTmqtt sub callback: /mqtt/test hello-rtthread
█
```



示例代码讲解

示例代码讲解

```
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include <rtthread.h>

#define DBG_ENABLE
#define DBG_SECTION_NAME "MQTT"
#define DBG_LEVEL DBG_LOG
#define DBG_COLOR
#include <rtdbg.h>

#include "paho_mqtt.h"
#define MQTT_URI "tcp://iot.eclipse.org:1883" // 配置测试服务器地址
#define MQTT_USERNAME "admin" //设置账号
#define MQTT_PASSWORD "admin" //设置密码
#define MQTT_SUBTOPIC "/mqtt/test" // 设置订阅主题
#define MQTT_PUBTOPIC "/mqtt/test" // 设置推送主题
#define MQTT_WILLMSG "Goodbye!" // 设置遗言消息

/* 定义 MQTT 客户端环境结构体 */
static MQTTClient client;
```

示例代码讲解

```
/* MQTT 订阅事件自定义回调函数 */
static void mqtt_sub_callback(MQTTClient *c, MessageData *msg_data)
{
    *((char *)msg_data->message->payload + msg_data->message->payloadlen) = '\0';
    LOG_D("mqtt sub callback: %.*s %.*s",
          msg_data->topicName->lenstring.len,
          msg_data->topicName->lenstring.data,
          msg_data->message->payloadlen,
          (char *)msg_data->message->payload);
    return;
}
/* MQTT 订阅事件默认回调函数 */
static void mqtt_sub_default_callback(MQTTClient *c, MessageData *msg_data)
{
    *((char *)msg_data->message->payload + msg_data->message->payloadlen) = '\0';
    LOG_D("mqtt sub default callback: %.*s %.*s",
          msg_data->topicName->lenstring.len,
          msg_data->topicName->lenstring.data,
          msg_data->message->payloadlen,
          (char *)msg_data->message->payload);
    return;
}
```

示例代码讲解

```
/* MQTT 连接事件回调函数 */
static void mqtt_connect_callback(MQTTClient *c)
{
    LOG_D("inter mqtt_connect_callback!");
}

/* MQTT 上线事件回调函数 */
static void mqtt_online_callback(MQTTClient *c)
{
    LOG_D("inter mqtt_online_callback!");
}

/* MQTT 下线事件回调函数 */
static void mqtt_offline_callback(MQTTClient *c)
{
    LOG_D("inter mqtt_offline_callback!");
}
```


示例代码讲解

```
/**
 * 这个函数创建并配置 MQTT 客户端。
 */
static void mq_start(void)
{
    /* 使用 MQTTPacket_connectData_initializer 初始化 condata 参数 */
    MQTTPacket_connectData condata = MQTTPacket_connectData_initializer;
    static char cid[20] = { 0 };

    static int is_started = 0;
    if (is_started)
    {
        return;
    }
    /* 配置 MQTT 结构体内容参数 */
    {
        client.uri = MQTT_URI;

        /* 产生随机的客户端 ID */
        rt_snprintf(cid, sizeof(cid), "rtthread%d", rt_tick_get());
    }
}
```

示例代码讲解

```
/* 配置连接参数 */
memcpy(&client.condata, &condata, sizeof(condata));
client.condata.clientID.cstring = cid;
client.condata.keepAliveInterval = 60;
client.condata.cleansession = 1;
client.condata.username.cstring = MQTT_USERNAME; //设置账号
client.condata.password.cstring = MQTT_PASSWORD; //设置密码

/* 配置 MQTT 遗嘱参数 */
client.condata.willFlag = 1;
client.condata.will.qos = 1;
client.condata.will.retained = 0;
client.condata.will.topicName.cstring = MQTT_PUBTOPIC; //设置推送主题
client.condata.will.message.cstring = MQTT_WILLMSG; //设置断开通知消息

/* 分配缓冲区 */
client.buf_size = client.readbuf_size = 1024;
client.buf = malloc(client.buf_size);
client.readbuf = malloc(client.readbuf_size);
if (!(client.buf && client.readbuf))
{
    LOG_E("no memory for MQTT client buffer!");
    goto _exit;
}
```

示例代码讲解

```
/* 设置事件回调函数 */
client.connect_callback = mqtt_connect_callback; //设置连接回调函数
client.online_callback = mqtt_online_callback; //设置上线回调函数
client.offline_callback = mqtt_offline_callback; //设置下线回调函数

/* 设置订阅表和事件回调函数*/
client.messageHandlers[0].topicFilter = MQTT_SUBTOPIC; //设置第一个订阅的 Topic
client.messageHandlers[0].callback = mqtt_sub_callback; //设置该订阅的回调函数
client.messageHandlers[0].qos = QOS1; //设置该订阅的消息等级

/* 设置默认的订阅主题*/
client.defaultMessageHandler = mqtt_sub_default_callback;
//设置一个默认的回调函数，如果有订阅的 Topic 没有设置回调函数，则使用该默认回调函数
}

/* 运行 MQTT 客户端 */
paho_mqtt_start(&client);
is_started = 1;

_exit:
return;
}
```

示例代码讲解

```
/**
 * 这个函数推送消息给特定的 MQTT 主题。
 *
 * @param send_str publish message
 *
 * @return none
 */
static void mq_publish(const char *send_str)
{
    MQTTMessage message;
    const char *msg_str = send_str;
    const char *topic = MQTT_PUBTOPIC; //设置指定 Topic
    message.qos = QOS1;                //消息等级
    message.retained = 0;
    message.payload = (void *)msg_str; //设置消息内容
    message.payloadlen = strlen(message.payload);

    MQTTPublish(&client, topic, &message); //开始向指定 Topic 推送消息

    return;
}
```

示例代码讲解

```
#ifndef RT_USING_FINSH
#include <finsh.h>
FINSH_FUNCTION_EXPORT(mq_start, startup mqtt client);
FINSH_FUNCTION_EXPORT(mq_publish, publish mqtt msg);
#ifdef FINSH_USING_MSH
MSH_CMD_EXPORT(mq_start, startup mqtt client);

int mq_pub(int argc, char **argv)
{
    if (argc != 2)
    {
        rt_kprintf("More than two input parameters err!!\n");
        return 0;
    }
    mq_publish(argv[1]);

    return 0;
}
MSH_CMD_EXPORT(mq_pub, publish mqtt msg);
#endif /* FINSH_USING_MSH */
#endif /* RT_USING_FINSH */
```



注意事项

注意事项

- 需要注意正确填写 **MQTT_USERNAME** 和 **MQTT_PASSWORD** ，如果 MQTT_USERNAME 和 MQTT_PASSWORD 填写错误，MQTT 客户端无法正确连接到 MQTT 服务器。
- 如果使用 MQTT TLS 加密连接，MQTT 线程栈至少需要 6144 字节。
- 电脑需要关闭防火墙

参考资料

- [源码](#)
- [MQTT 官网](#)
- [Paho 官网](#)
- [IBM MQTT 介绍](#)
- [Eclipse paho.mqtt 源码](#)