道哥的第 022 篇原创

一、前言

在上一篇文章中物联网网关开发：基于MQTT消息总线的设计过程(上)，我们聊了在一个物联网系统的网关中，如何利用 MQTT 消息总线，在嵌入式系统内部实现多个进程之间的相互通信问题。

这个通信模型的最大几个优点是：

1. 模块之间解耦合;
2. 各模块之间可以并行开发;
3. 把 TCP 链接和粘包问题交给消息总线处理，我们只需要处理业务层的东西;
4. 调试方便;

以上只是描述了在一个嵌入式系统内部，进程之间的通信方式，那么网关如何与云平台进行交互呢?

在上一篇文章中已经提到过：网关与云平台之间的通信方式一般都是客户指定的，就那么几种(阿里云、华为云、腾讯云、亚马逊AWS平台)。一般都要求网关与云平台之间处于长连接的状态，这样云端的各种指令就可以随时发送到网关。

这一篇文章，我们就来聊一聊这部分内容。

在公众号回复：mqtt，获取示例代码的网盘地址。

二、与云平台之间的 MQTT 连接

目前的几大物联网云平台，都提供了不同的接入方式。对于网关来说，应用最多的就是 MQTT 接入。

我们知道，MQTT 只是一个协议而已，不同的编程语言中都有实现，在 C 语言中也有好几个实现。

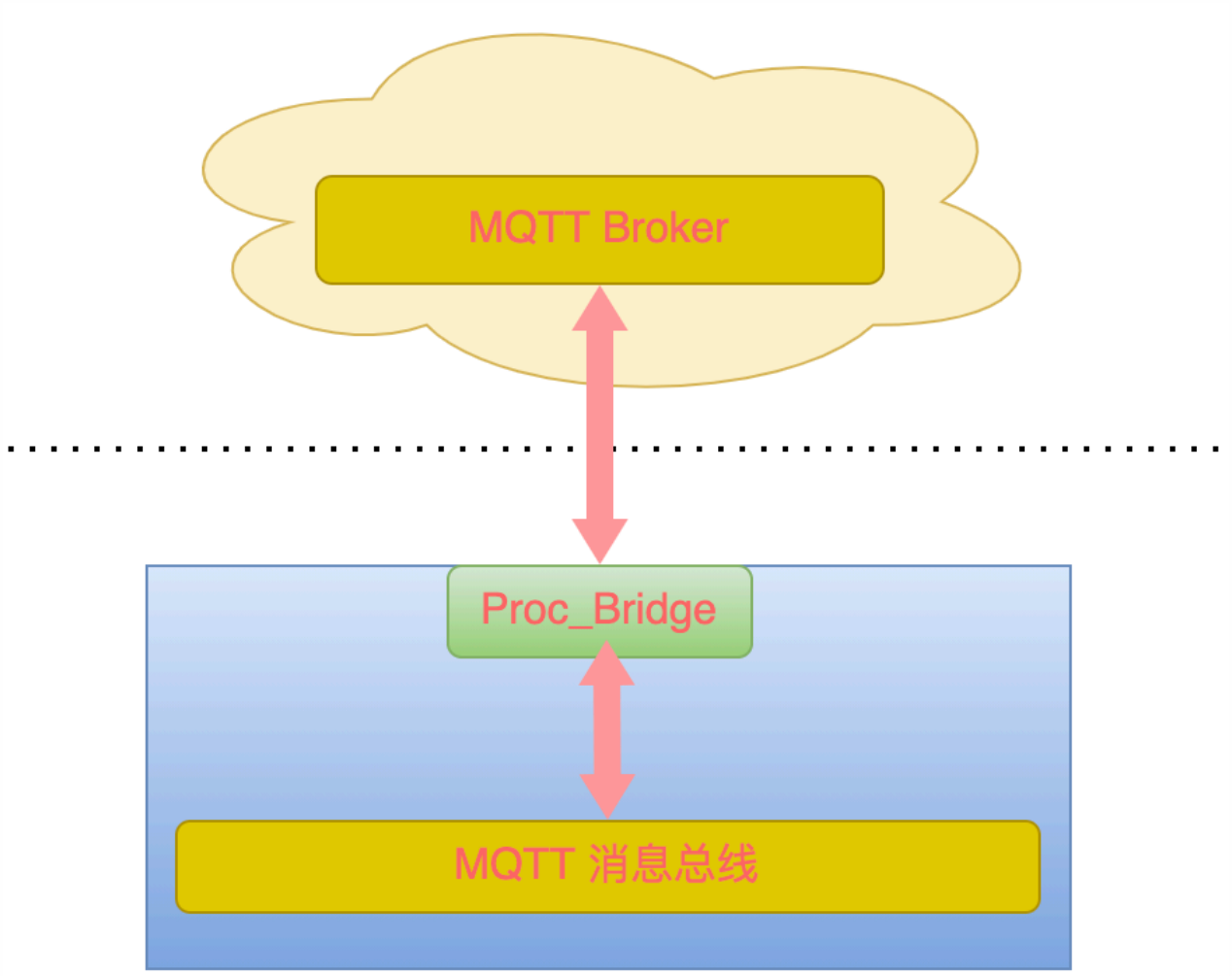
在网关内部，运行着一个后台 deamon: MQTT Broker，其实就是 mosquitto 这个可执行程序，它充当着消息总线的功能。这里请大家注意：因为这个消息总线是运行在嵌入式系统的内部，接入总线的客户端就是需要相互通信的那些进程。这些进程的数量是有限的，即使是一个比较复杂的系统，最多十几个进程也就差不多了。因此，mosquitto 这个实现是完全可以支撑系统负载的。

那么，如果在云端部署一个 MQTT Broker，理论上是可以直接使用 mosquitto 这个实现来作为消息总线的，但是你要评估接入的客户端(也就是网关)在一个什么样的数量级，考虑到并发的问题，一定要做压力测试。

对于后台开发，我的经验不多，不敢(也不能)多言，误导大家就罪过了。不过，对于一般的学习和测试来说，在云端直接部署 mosquitto 作为消息总线，是没有问题的。

三、Proc_Bridge 进程：外部和内部消息总线之间的桥接器

下面这张图，说明了 Proc_Bridge 进程在这个模型中的作用：



> 1. 从云平台消息总线接收到的消息，需要转发到内部的消息总线；
> 2. 从内部消息总线接收到的消息，需要转发到云平台的消息总线；

如果用 mosquitto 来实现，应该如何来实现呢？

1. mosquitto 的 API 接口

mosquitto 这个实现是基于回调函数的机制来运行的，例如：

```
// 连接成功时的回调函数
```

```
void my_connect_callback(struct mosquitto *mosq, void *obj, int rc)
{
    // ...
}

// 连接失败时的回调函数
void my_disconnect_callback(struct mosquitto *mosq, void *obj, int result)
{
    // ...
}

// 接收到消息时的回调函数
void my_message_callback(struct mosquitto *mosq, void *obj, const struct
mosquitto_message *message)
{
  // ..
}

int main()
{
    // 其他代码
    // ...

    // 创建一个 mosquitto 对象
    struct mosquitto g_mosq = mosquitto_new("client_name", true, NULL);

    // 注册回调函数
    mosquitto_connect_callback_set(g_mosq, my_connect_callback);
  mosquitto_disconnect_callback_set(g_mosq, my_disconnect_callback);
  mosquitto_message_callback_set(g_mosq, my_message_callback);
  // 这里还有其他的回调函数设置

  // 开始连接到消息总线
  mosquitto_connect(g_mosq, "127.0.0.1", 1883, 60);

  while(1)
  {
    int rc = mosquitto_loop(g_mosq, -1, 1);
    if (rc) {
      printf("mqtt_portal: mosquitto_loop rc = %d \n", rc);
      sleep(1);
      mosquitto_reconnect(g_mosq);
    }
  }

  mosquitto_destroy(g_mosq);
  mosquitto_lib_cleanup();
  return 0;
}
```

以上代码就是一个 mosquitto 客户端的最简代码了，使用回调函数的机制，让程序的开发非常简单。

mosquitto 把底层的细节问题都帮助我们处理了，只要我们注册的函数被调用了，就说明发生了我们感兴趣的事件。

这样的回调机制在各种开源软件中使用的比较多，比如：glib 里的定时器、libevent通讯处理、libmodbus 里的数据处理、linux 内核中的驱动开发和定时器，都是这个套路，一通百通！

在网关中的每个进程，只需要添加上面这部分代码，就可以挂载到消息总线上，从而可以与其它进程进行收发数据了。

2. 利用 UserData 指针，实现多个 MQTT 连接

上面的实例仅仅是连接到一个消息总线上，对于一个普通的进程来说，达到了通信的目的。

但是对于 Proc_Bridge 进程来说，还没有达到目的，因为这个进程处于桥接的位置，需要同时连接到远程和本地这两个消息总线上。那么应该如何实现呢？

看一下 mosquitto_new 这个函数的签名：

```
/*
 * obj - A user pointer that will be passed as an argument to any
 *       callbacks that are specified.
*/
libmosq_EXPORT struct mosquitto *mosquitto_new(const char *id, bool
clean_session, void *obj);
```

最后一个参数的作用是：可以设置一个用户自己的数据(作为指针传入)，那么 mosquitto 在回调我们的注册的任何一个函数时，都会把这个指针传入。因此，我们可以利用这个参数来区分这个连接是远程连接？还是本地连接。

所以，我们可以定义一个结构体变量，把一个 MQTT 连接的所有信息都记录在这里，然后注册给 mosquitto。当 mosquitto 回调函数时，把这个结构体变量的指针回传给我们，这样就拿到了这个连接的所有数据，在某种程度上来说，这也是一种面向对象的思想。

```
// 从来表示一个 MQTT 连接的结构体
typedef struct{
  char *id;
  char *name;
  char *pw;
  char *host;
  int port;
  pthread_t tHandle;
  struct mosquitto *mosq;
  int mqtt_num;
}MQData;
```

完整的代码已经放到网盘里了，为了让你先从原理上看明白，我把关键几个地方的代码贴在这里：

```
// 分配结构体变量
MQData userData = (MQData *)malloc(sizeof(MQData));

// 设置属于这里连接的参数: id, name 等等
```

```
// 创建 mosquitto 对象时，传入 userData。
struct mosquitto *mosq = mosquitto_new(userData->id, true, userData);

// 在回调函数中，把 obj 指针前转成 MQData 指针
static void messageCB(struct mosquitto *mosq, void *obj, const struct
mosquitto_message *message)
{
  MQData *userData = (MQData *)obj;

  // 此时就可以根据 userData 指针中的内容分辨出这是哪一个链接了
}
```

另外一个问题：不知道你是否注意到示例中的 mosquitto_loop() 这个函数？这个函数需要放在 while 死循环中不停的调用，才能出发 mosuiqtto 内部的事件。(其实在 mosuiqtto 中，还提供了另一个简化的函数 mosquitto_loop_forever)。

也就是说：在每个连接中，需要持续的触发 mosquitto 底层的事件，才能让消息系统顺利的收发。因此，在示例代码中，使用两个线程分别连接到云平台的总线和内部的总线。

四、总结

经过这两篇文章，基本上把一个物联网系统的网关中，最基本的通信模型聊完了，相当于是一个程序的骨架吧，剩下的事情就是处理业务层的细节问题了。

万里长征，这才是第一步！

对于一个网关来说，还有其他更多的问题需要处理，比如：MQTT 连接的鉴权(用户名+密码，证书)、通信数据的序列化和反序列化、加密和解密等等，以后慢慢聊吧，希望我们一路前行！

在公众号回复：mqtt，获取示例代码的网盘地址。

不吹嘘，不炒作，不浮夸，认真写好每一篇文章！

欢迎转发、分享给身边的技术朋友，道哥在此表示衷心的感谢！

推荐阅读

我最喜欢的进程之间通信方式-消息总线
C语言指针-从底层原理到花式技巧，用图文和代码帮你讲解透彻
一步步分析-如何用C实现面向对象编程
提高代码逼格的利器：宏定义-从入门到放弃
原来gdb的底层调试原理这么简单
利用C语言中的setjmp和longjmp，来实现异常捕获和协程

关于加密、证书的那些事
深入LUA脚本语言，让你彻底明白调试原理