# 消息队列设计

SCMT项目中的消息设计使用了ZMQ库的C#版本。ZMQ提供了几种协议，常用的有IPC和TCP。IPC协议用于进程间消息，TCP可用于本地，也可用于网络间的消息传输。

消息队列的经典模式有：publish-subscribe模式，request-response模式和pull-push模式。pub-sub模式是异步模式，publisher发布消息，subscriber订阅消息；req-res模式是同步模式，客户端发送request后，会一致阻塞到response server返回结果；pull-push模式是异步模式。在SCMT项目中，用到的模式有pub-sub模式和request-response模式。

SCMT项目中pub-sub模式使用了TCP协议。publisher和subscriber是成对出现的。比如模块A使用TCP协议在1234端口发布消息，那么模块B要订阅模块A发布的消息，就需要连接到1234端口上。每个模块可以订阅多个连接的消息，也可以发布消息到多个连接。

## pub-sub模式

**publisher类的构造函数原型如下：**

public PublisherClient(int port, string addr = "127.0.0.1")

port表示发布消息到哪个端口，addr表示服务器的地址，一般默认是127.0.0.1。

发布消息重载了多个原型：

public void PublishMsg(string topic, byte[] msg) //发布的数据是字节数组

public void PublishMsg(string topic, string msg) //发布的数据是字符串

两个原型都要传入topic。topic具有特别的意义，subscriber订阅相同的topic，才能进行数据的过滤。topic的管理当前没有好的设计方案，暂时只统计了每个模块的信息，后面可以设计一个专门的配置文件用于topic的管理

**subscriber类的构造函数原型如下：**

public SubscribeClient(int port, string addr = "127.0.0.1")

订阅消息函数原型：

public bool AddSubscribeTopic(string topic, HandlerSubscribeMsg handler)

topic需要和publisher保持一致，handler是处理该topic的回调函数。HandlerSubscribeMsg是一个委托，声明如下：

public delegate void HandlerSubscribeMsg(SubscribeMsg msg);

而其中SubscribeMsg结构用于接收publisher发布的数据，定义如下：

public class SubscribeMsg

{

public string Topic { get; }

public byte[] Data { get; }

public SubscribeMsg(byte[] data, string topic)

{

Topic = topic;

Data = data;

}

}

取消订阅函数原型：

public bool CancelSubscribeTopic(string topic)

如果不想再处理某些topic的数据，就调用这个函数取消订阅。

subscriber初始化完成后，需要调用Run()函数接收数据。

## request-respon模式

未实现。

# ATP相关模式设计

1. LinkMgr模块

管理ATP连接的创建和断开，提供接口：

public bool ConnectNetElement(string ip, NetElementConfig neConfig)

public bool DisconnectNe(string ip)

LinkMgr模块订阅topic：

udp-recv://板卡地址:5002

1. SessionMgr模块