**现场端与中心代理交互设计**

问题描述：通信交互过程是指具体协议中规定的现场机与中心代理之间的数据收发过程。

问题分析：对于不同的协议，具体任务的交互过程也随之改变，因此，通信交互方案应具备一定的稳定性，实现协议可变更，具有通用性。

由于现场机与中心代理连接过程中，数据发送的随机性，不能确定某时刻接收的数据是否为本次应接收的数据，因此，必须解决每一次交互任务中，发送数据后，如何接收正确的响应帧。

解决方案：将命令交互的过程分成异步处理，发送请求后，异步处理应答等接受的数据，对于接收到的数据进行统一处理。

异步处理时，按照212协议命令进行分类，每条命令对应一个处理过程，每条命令的处理过程分为几个阶段（应答，接收数据，执行结果等等）等所有阶段处理完毕则处理过程结束。



中心代理与现场机交互的处理上整体上分为三部分功能：

（1）接口数据处理：分类处理通过功能接口传来的任务，对每一类创建对应的处理过程processor，通过维护一个队列循环处理传来的任务；

（2）Processor管理：通过维护一个processor列表来管理多个processor的加入和移除。

（3）数据接收处理：从socket通信模块接收数据，并将接收的数据分发给维护的processor进行处理。

Processor完成具体的处理任务，包括数据解析入库等处理。每个任务命令对应一个processor处理过程，提供具体的发送请求接口与处理接口。并使用运行状态字来标识处理的阶段，并完成各个阶段的具体的处理过程（解析，入库等）。

有请求时调用发送请求接口，在其内部调用通信接口发送数据到现场机，然后等待数据响应，一旦有数据，调用数据处理接口，在其内部完成数据各阶段的具体处理实现。

由于processor的种类较多，每次处理时，发送给全部的processor，让每个processor自行判断是否该处理较为浪费，因此，对此处设计进行优化，将数据接收处理与processor设计成观察者模式。

在观察者模式中，processor为观察者，数据接收处理中心为被观察者，在创建processor的时候主动将自己注册到processor列表中，数据接收处理中心一点收到数据，就将数据通知给注册了的processor，由这些processor自行判断处理，等processor的所有过程处理完后，主动从processor列表中注销自己，不再接收数据。

方案优点：

（1）通用通信处理框架，异步处理的过程使得协议可自定义（目前使用的212协议也可以兼容2005版本与2017版本）；

（2）所有接收数据进行统一处理，对于异常信息只是作为其中一类进行处理，使异常情况的处理变得更简单。

提供统一的对外接口，外部调用该接口，通过参数传递具体任务命令，内部对这些任务进行分类处理。

**Processor**

每条命令用一个processor实现，提供具体的发送请求与处理接口，使用运行状态字来标识处理的阶段，并完成各个阶段的具体的处理过程（解析，入库等）。

Send()发送请求接口，内部调用通信接口

Handle()数据处理接口

Processor的分类：

上传类：（不发送，只接收，直接接收数据处理，如报警信息、上传数据）

通知类：（发送后，只接收应答）

请求类：又分为

设置类（发送后，接收应答、接收执行结果）

获取类（发送后，接收应答、接收获取的数据（一次或多次）、接收执行结果）

注：实时数据较为特殊，具体实现时需要特殊处理

**Proxy**

proxy分类处理通过统一功能接口传来的任务，对每一类创建对应的处理过程processor，通过维护一个队列循环处理；

Proxy通过维护一个processor列表来管理多个processor；

接收数据，并将接收的数据分发给维护的processor进行处理

从socket流中读取数据，按照协议进行匹配，将符合协议的数据帧存入缓存队列，然后循环处理缓存队列中的数据帧，将其分发给processor列表中的processor根据自身状态判断是否处理或者如何处理。

由于processor的种类较多，每次处理时，发送给全部的processor，让每个processor自行判断是否该处理较为浪费，因此，对此处设计进行优化。

将远程端主动发送过来的

观察者模式

依据

同步接收时，接收的数据可能是其他类的有用数据，或者无用数据，但在超时时间内，远程机确实已经发送成功了对应所需的数据，对于异常的处理较为复杂；且摆脱不了某个时刻要判断是主动状态还是被动状态以及主被动状态转换的复杂性。

**优点：**

（1）通用通信处理框架，异步处理的过程使得协议可自定义（目前使用的212协议也可以兼容2005与2017）；

（2）所有接收数据进行统一处理，对于异常信息只是作为其中一类进行处理，使异常情况的处理变得更简单。