使用netty实现根据自定义协议格式进行socket通信，协议格式定义如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Header | 版本 | Cmd | Code(单位标识) | 是否加密 | 内容的长度 |
| 1字节 | 1字节 | 1字节 | 4字节 | 1字节 | 8字节 |
| 具体的内容 | | | | | |
| Content | | | | | |

Header：默认为0x82

Version：根据软件实际情况定义，0x10：表示1.0版本

Cmd：表示具体的请求类型，程序要根据cmd的不同类型，做不同的逻辑处理

unitcode（单位标识）：根据具体的单位信息填写，占用4个字节

是否加密：

0 1 2 3 4 5 6 7

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |

第2位表示是否加密，1：表示加密，0：表示不加密，如果第2位为0，无效考虑后面的位；

第3位表示加密算法的类型，0：表示对此加密，1：表示非对称加密

第4、5、6、7四位表示加密的算法

**对称加密算法**

1：表示AES，2：表示DES，3：表示3DES

**非对称加密算法**

6：RSA、7：DSA、8：ECC

数据请求流程：

客户端发起请求，请求数据格式为自定义的协议格式，服务端接收到请求后首先解析协议数据，根据不同的cmd做不同的业务流程处理。

例如：cmd为0x11：服务端要将某数据库表的数据（数据量很大，需要分批发送），封装成协议数据后返回给客户端（mysql数据库表结构可以自己随便定义），客户端收到数据后将数据写入到另一张mysql数据库表中。

Cmd为0x12: 客户端将大量数据（需要分批发送）封装成协议格式后发送给服务端，服务端收到数据后，将数据写入到数据库表中，并返回那些数据写入成功，那些数据写入失败（返回成功失败数据的ID即可）