详细设计

1. 消息体设计

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 |  |  |  |
| Message | String | 消息内容 |  |
| Type | String | 消息类型  0 普通聊天消息   1. 加好友消息 2. 点赞消息 3. 群组消息 4. 整点消息 5. 好友通过消息 |  |
| From | String | 消息来源（  Type=0的时候就是发送方code  Typ=3的时候是群组id  其他都是system  ） |  |
| Realyfrom | String | 消息真实发送者  当type=1,2,3,5的时候，储存消息真正来源。 |  |
|  |  |  |  |

1. 消息传输设计
   1. 点对点聊天文本消息

使用kafka进行传递，每个客户端订阅自身的主题，(topic命名规则:msg\_usercode)，创建topic，在登录的时候由服务器来创建，客户端只能订阅，不能创建。

* 1. 加好友设计

A用户向服务器发送加好友请求，服务器发送加好友消息（type=1，来源为system）（topic命名规则:msg\_usercode）给B用户,**同时**服务器在redis里存储这个好友请求的消息（key命名规则：Ausercode\_friend\_Busercode），B用户通过后，向服务器发送好友通过请求，服务器判断redis KEY是否存在，存在就发送好友通过消息（type=5，来源system）给A用户，同时删除key以及A,B用户好友列表互加上对方。A客户端收到后，更新客户端好友列表。

2.3点赞设计

A用户向服务器发送点赞请求，服务器器发送点赞消息（type=2，来源为system）（topic命名规则:msg\_usercode）给B用户

2．4群组消息

用户加入群组后，订阅群组消息，（topic 命名规则：group\_id）（type=3）。

1. Api设计

待补充

1. 点赞功能设计

点赞使用redis 自带的incr 功能实现，key为 usercode\_good，类型为数字型，服务端使用cron每10分钟 更新所有点赞数到数据库中。每个用户只可点击一次。（低优先级）

1. 心跳包设计

心跳包使用redis 的key过期机制来实现，登录的时候创建登录key(命名规则usercode\_login)设置过期时间30秒，客户端每隔20秒发送一次在线请求到服务器。服务器更新登录key的过期时间。

服务器接收心跳包后，放回好友列表和在线状态。通过这个机制来更新好友列表信息和状态。

服务端代码 注册 KeyExpirationEventMessageListener 事件，这样可以第一时间知道redis过期，同时开启离线数据缓存。

1. 离线消息存储设计

6.1 点对点聊天信息，点赞，加好友信息

客户端掉线或者下线以后，服务端订阅下线用户的主题（topic命名规则:msg\_usercode），保存到BU\_MESSAGE 表中。

6.2群组消息

服务器需要订阅所有的群组消息，收到群组消息后，判断每个用户是否在线，不在线的用户 ，保存到BU\_MESSAGE 表中。

1. 系统模块划分



8 类图

 

 

9．系统初始化

系统重启后，需要在kafka上创建所有用户的topic,并对所有用户的topic进行监听，直到有用户上线后，开始取消对应用户的监听。