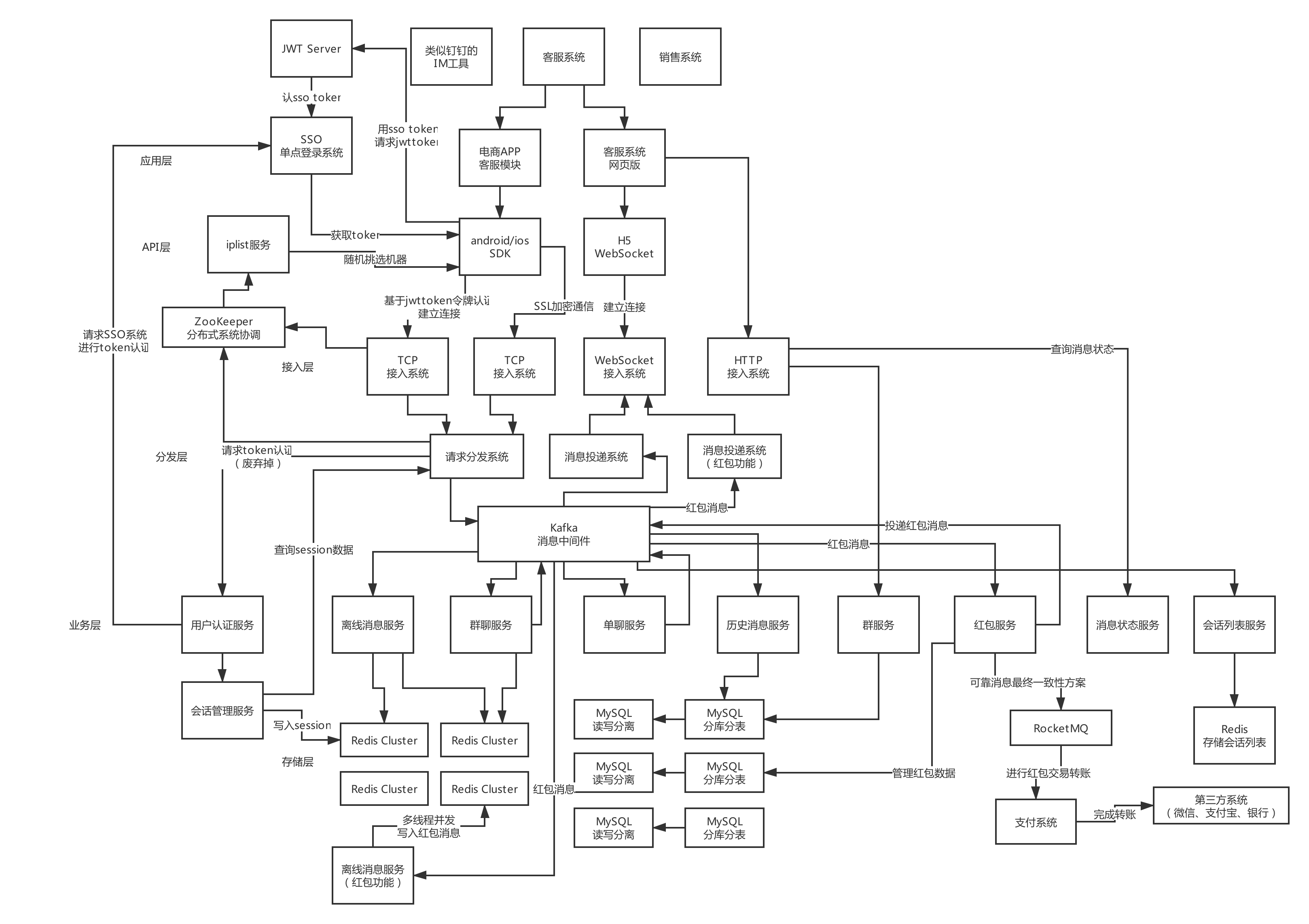
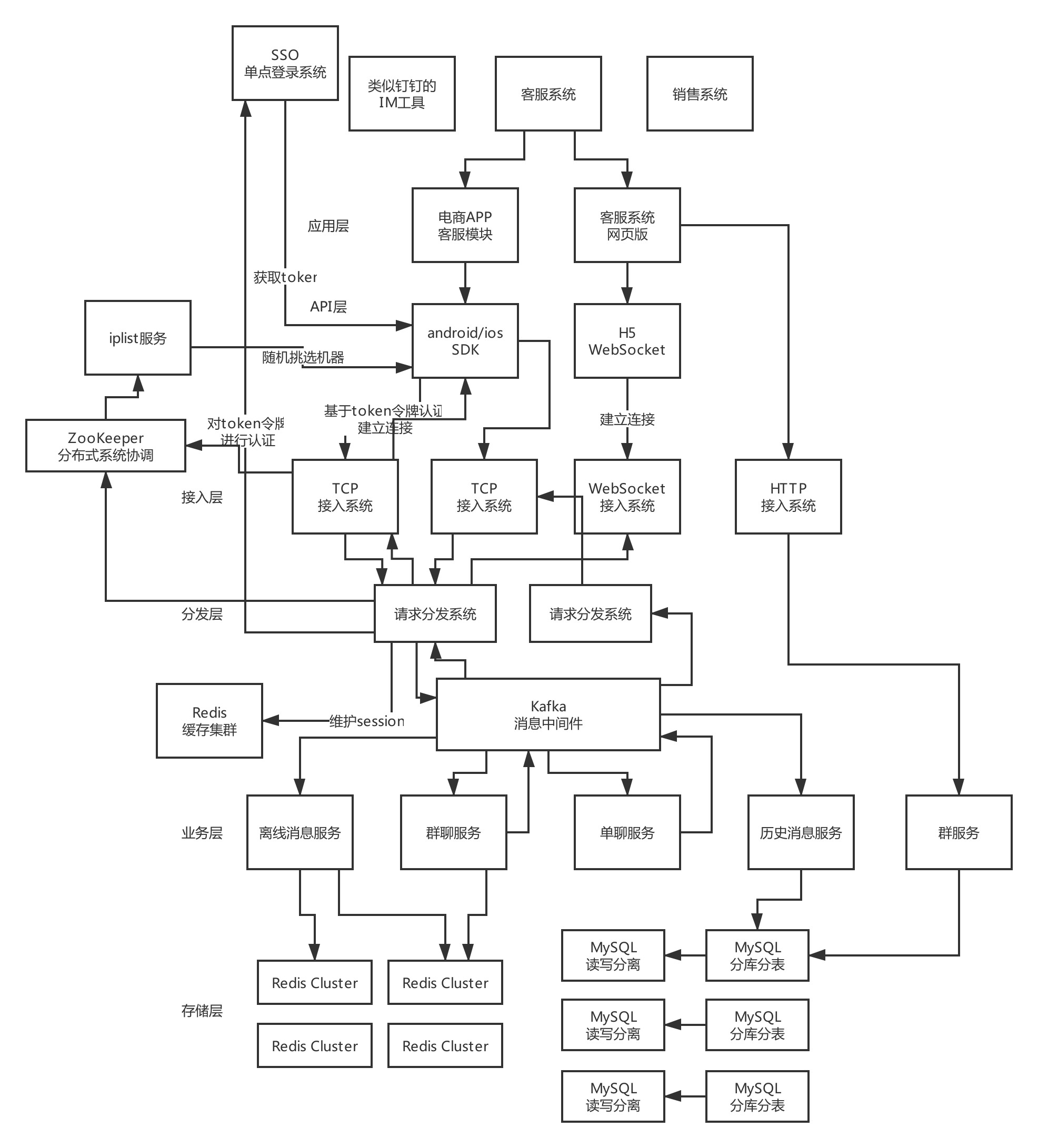
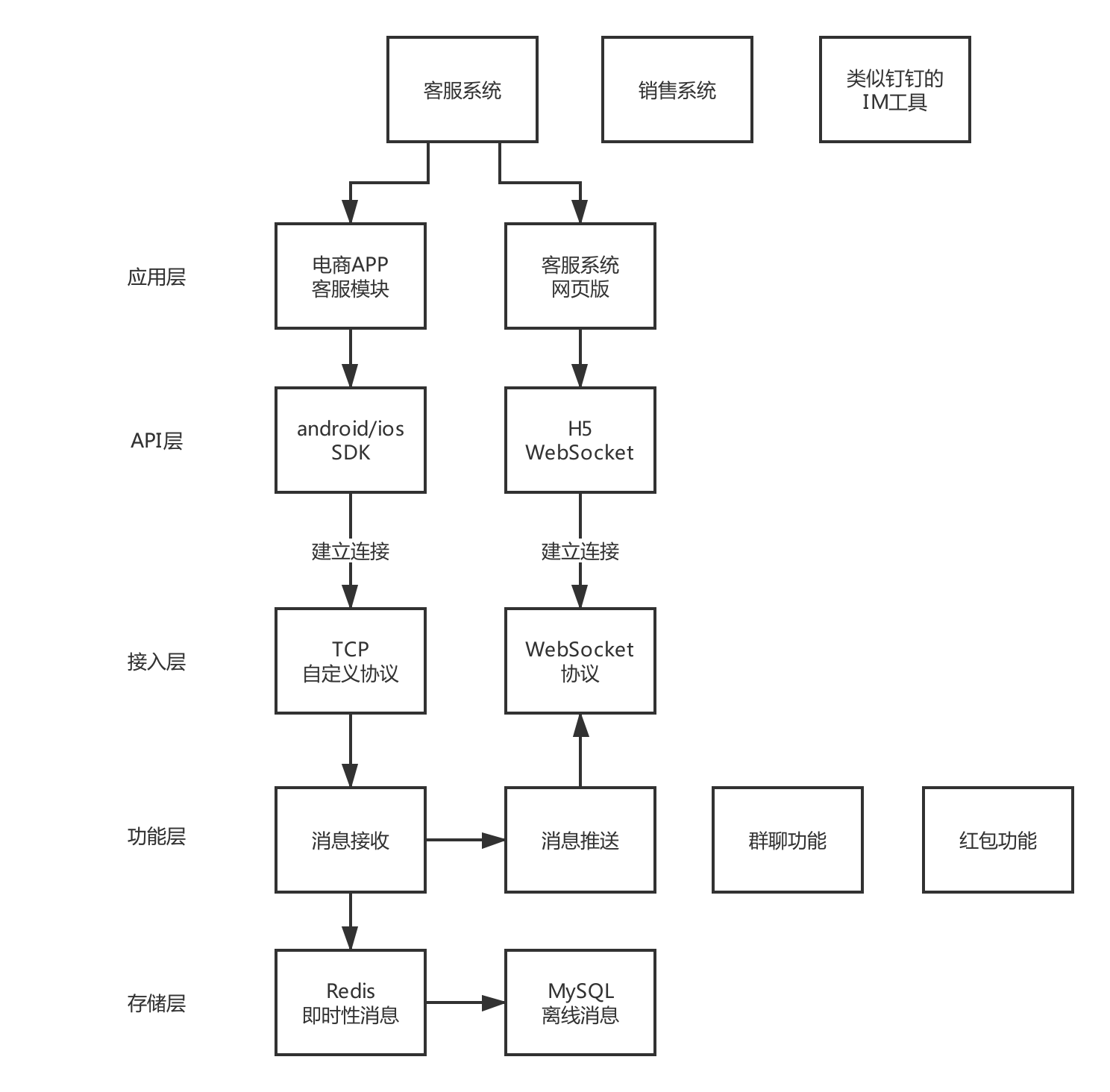
# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/123\_学完截止目前的IM课程后还缺了哪些东西以及最后一周IM课程预告/01\_IM系统架构设计-2.png**



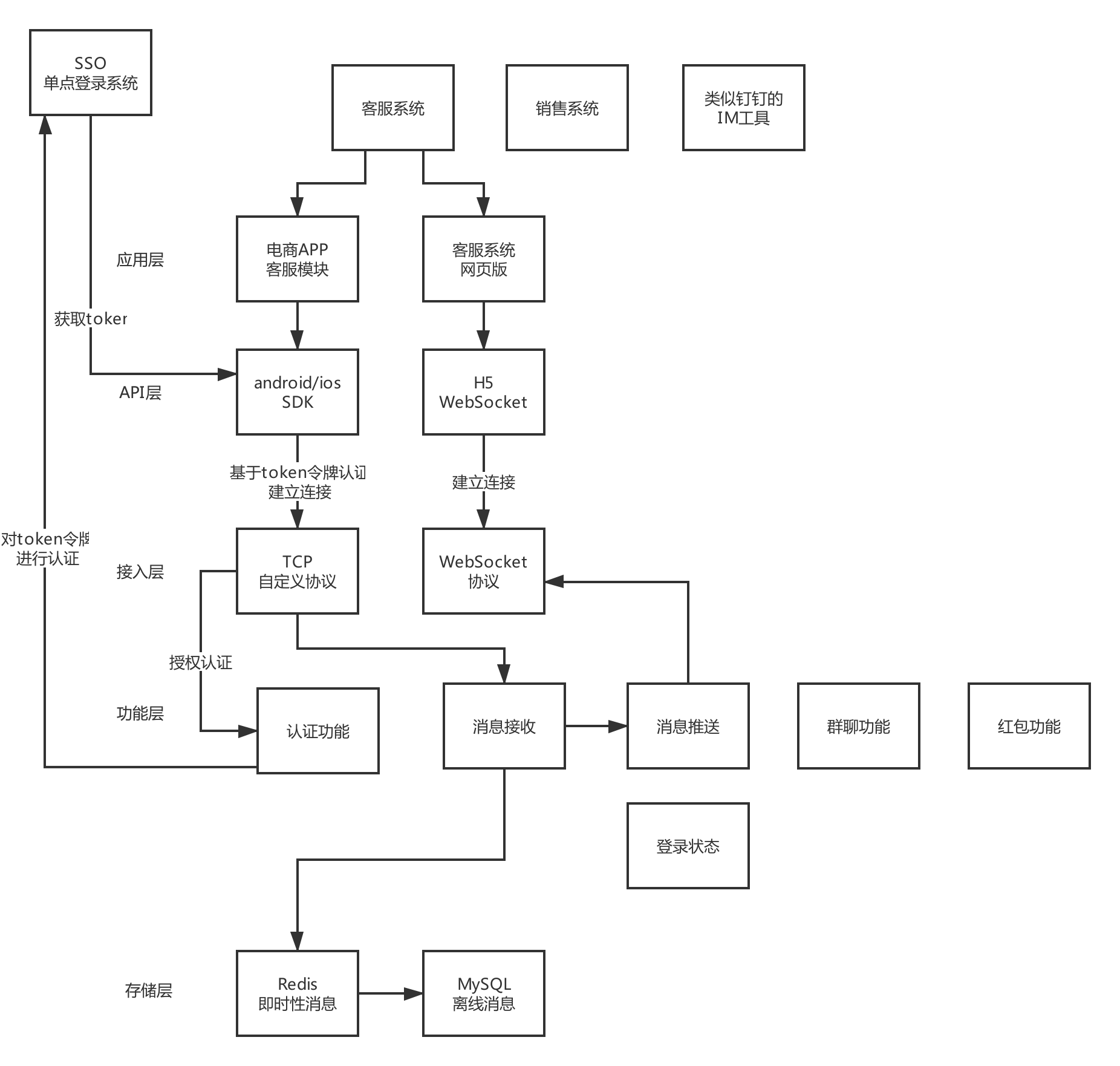
# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/103\_接下来IM系统的架构设计 还有哪些东西要讲解？/01\_IM系统架构设计-2.png**



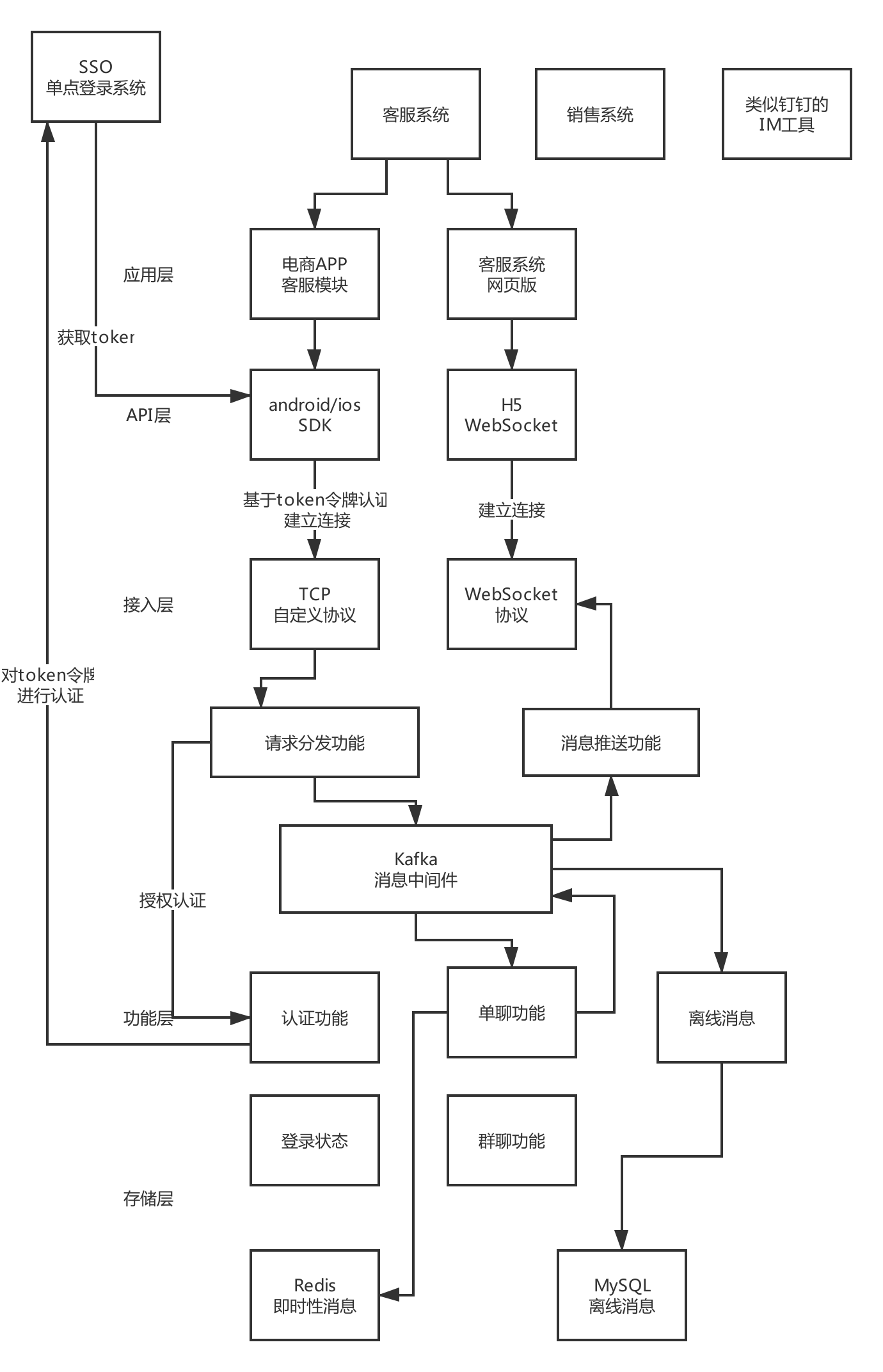
# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/001~007资料/002\_一个IM系统的架构大致包含哪些东西？/01\_IM系统架构设计.png**



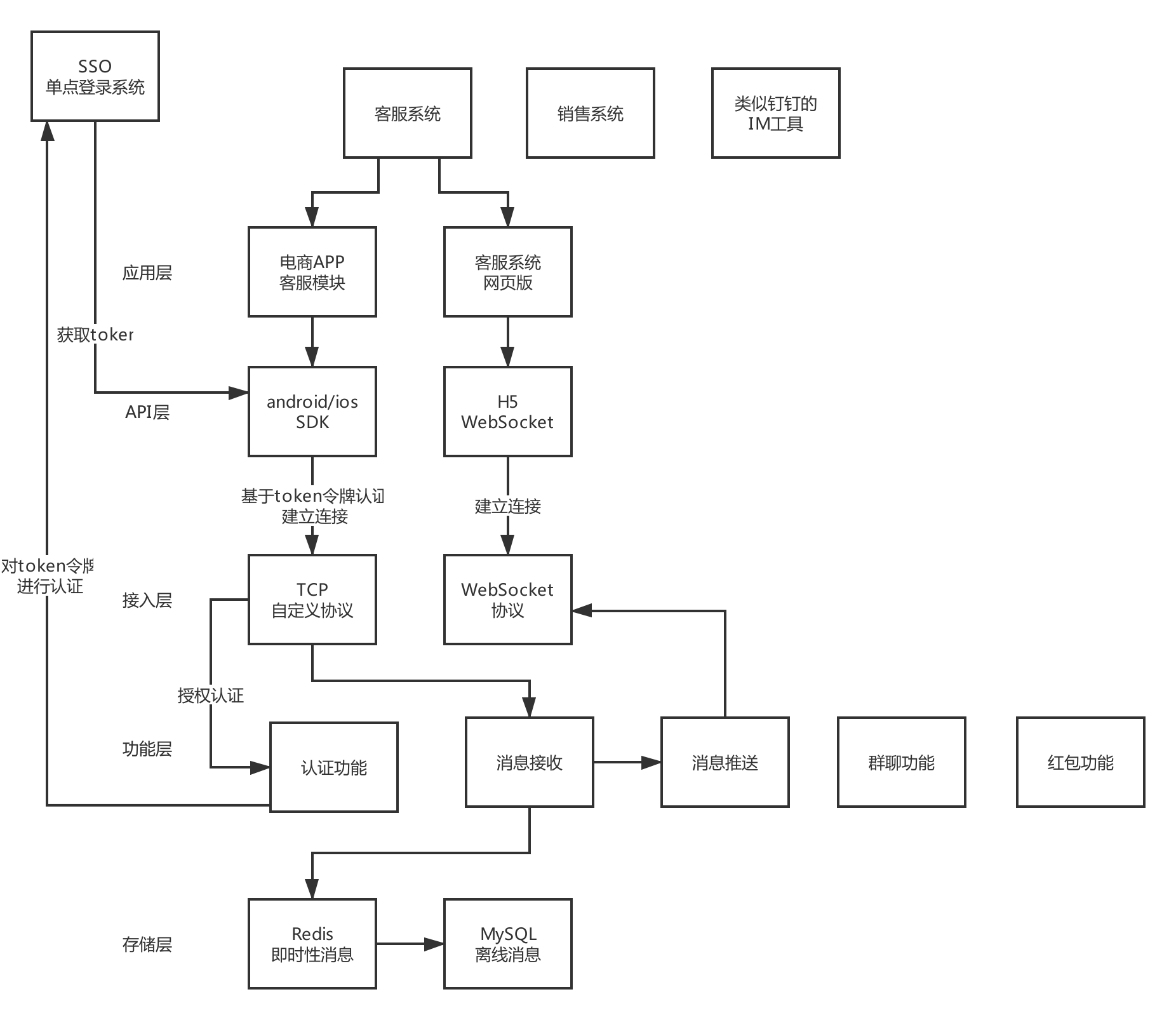
# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/001~007资料/004\_客户端关闭、重复登录的时候应该如何处理？/01\_IM系统架构设计.png**



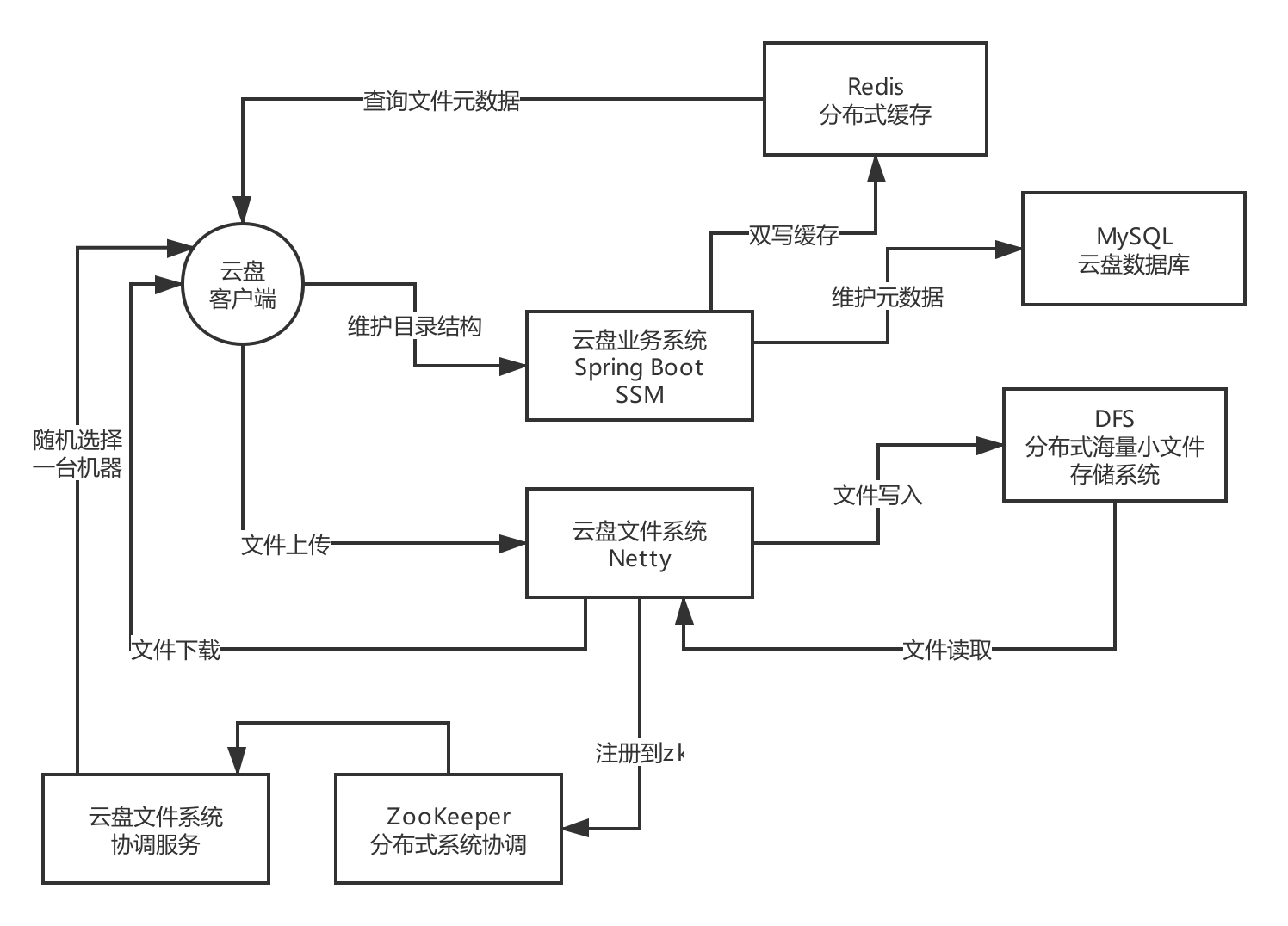
# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/001~007资料/005\_如何使用消息中间件让IM系统的功能模块实现解耦？/01\_IM系统架构设计.png**



# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/001~007资料/003\_IM系统的授权认证应该怎么来做？/01\_IM系统架构设计.png**



# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/15\_Netty各种功能在三个项目中落地实战的介绍/01\_比较low的云盘系统架构.png**



# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/045~055资料/045\_为IM系统部署一个示范用的单机版Redis/笔记 .docx**

在十一之前，已经搞定了基本的IM系统网络通信那块的代码，那块东西是最核心的一些东西，第一大块是完成核心的业务功能，单聊，群聊，历史消息；第二大块IM系统特殊的一些东西，消息的顺序、timeline模型；第三大块，架构上的东西，缓存、消息（Kafka）、分库分表，大数据HBase

《045\_为IM系统部署一个示范用的单机版Redis》

从redis官网下载 redis-5.0.5.tar.gz，然后解压缩

sudo make test

sudo make install

redis-server

redis-cli

测试一下，应该没问题

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/14\_作业：分析一下自己公司的业务有哪些可以套用DFS技术的/笔记.doc**

dfs系统，文件后端系统，跟你自己公司的业务深度的融合

公司的核心业务，有哪些业务是跟文件相关联的，上传下载文件的，梳理一下你们的文件上传和下载业务的需求，功能点，dfs，开发一个文件后端系统，业务给实现一下

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/01\_仿钉钉云盘的文件分享平时是怎么玩儿的？/笔记.doc**

Netty，核心功能已经精讲完毕，里面源码细节，提了一下思路，给大家留了作业，让大家自己去扣里面的源码细节

鱼，渔

云盘项目（把Netty和DFS），IM项目，消息推送项目

云盘项目，最最核心的一点其实是在分布式文件存储，DFS占据了这个项目的95%内容，基于Netty封装一个云盘的后端系统，5%

钉钉，QQ

钉钉，下载一个，国内各大中小公司，公司内部同事沟通，都是走钉钉

比如你有一个文件，拖拽到钉钉的一个云盘区域里，在云盘里可以创建一些目录，文件就直接上传到了你的目录里去了

你自己可以下载这个文件，其实你也可以把这个文件分享给其他的同事，如果其他同事看到这个分享之后，就可以下载这个文件

APP、系统里都有，电商公司里，后台编辑这个商品详情页，插入一些图片，在后台系统里有一个商品素材库，在公共素材库上传一个图片，然后把这个图片分享给其他的同事，也可以在编辑商品详情的时候去插入一个图片进去

社交的APP，你可以在自己私有的区域里上传一些自己的日常生活照片，你还可以把这些东西一键分享给其他人去看

DFS系统，这个系统什么时候用呢？在哪里用呢？

钉钉云盘，文件的上传和分享，下载，就可以基于DFS来做

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/06\_思考一下：这个架构中是否需要处理文件的粘包拆包？/笔记.doc**

基于Netty客户端和服务端在进行文件传输的过程中，是不是也要进行粘包和拆包的处理？要保证说你的一个文件是传输完整的

Netty处理粘包和拆包，后面开发的时候，我来给大家讲解一下

在发送数据的时候，你在一个文件数据末尾，可以加入一个特殊符号，nettyserver端在接收文件的时候，他会按照你指定的特殊符号来接收文件，传递给你的是一个完整的一个文件，不需要你自己手动在底层做过多的网络开发

dubbo源码，序列化，netty后面两个项目，也会讲解序列化和反序列化，定制协议

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/10\_对架构的思考：如何保证文件传输的高可用性？/笔记.doc**

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/13\_作业：自己玩儿一下钉钉的云盘分享功能，梳理功能点/笔记.doc**

云盘，钉钉里面就有云盘，创建目录，拖拽文件就可以上传，分享这个文件给你的同事，功能点都梳理出来，共享交流的精神，作业，可以分享到你所在的学员群里去，跟大家一起来共享，交流，针对作业，可以跟其他学员一起交流

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/15\_Netty各种功能在三个项目中落地实战的介绍/笔记.doc**

netty一共是规划了三个项目

第一个云盘项目，代码都不会写多少的，留下来作业，让你自己去完成这个项目，95%的核心的点，都在之前的dfs里

netty粘包和拆包的处理，留作业让大家自己去动手开发一个云盘文件系统，跟dfs进行交互，把dfs落地到业务场景里去

云盘业务系统，缓存、数据库，自己设计表，都给实现了

有问题来问我啊

netty的一个功能，拆包和粘包，20mb以内的小文件

第二个项目，IM项目，IM系统，复杂架构设计，Netty在里面的核心功能的开发，核心代码的开发，把非核心的，部分核心的，当做作业留给大家自己去实现，还会留作业给大家，去看看自己公司业务里有哪些是IM的场景

第三个项目，消息推送项目，复杂架构设计，Netty开发里面核心的代码

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/08\_对架构的思考：如何保证文件传输的高性能？/笔记.doc**

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/07\_思考一下：文件传输过程中的异常是如何处理的？/笔记.doc**

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/11\_对架构的思考：如何保证系统全链路是可伸缩的？/笔记.doc**

你的Netty云盘文件系统，他们在每次在一台机器上启动的时候，都可以把自己的地址写入到一个zk的地方去

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/05\_另外一种架构优化方案：全链路基于内存传输数据/笔记.doc**

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/02\_一个比较low的云盘系统架构是如何设计的/笔记.doc**

接收文件，元数据写入MySQL数据库，包括谁上传的，上传时间之类的，属于哪个文件夹，文件夹目录树可以在MySQL里用父子关系存储，总之云盘的元数据都在数据库里存储，这里基于数据库可以玩很多花样

比如文件目录的权限，某个目录哪些人是有权限可以看到的

比如可能某人没权限看到一个文件，但是你分享给他之后，他就可以看到了

这些常规性的云盘功能，都基于数据库来实现即可

上传的文件先写入本地磁盘文件，然后将本地磁盘文件上传到dfs上去，才算上传结束

读文件的时候，都直接从dfs上去读，因为dfs提供多副本机制，保证冗余，保证文件是安全的

很多码农，先写入本地磁盘，然后把本地磁盘里的文件重新读取出来，用DFSClient上传到DFS上去

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/03\_这么low的云盘系统架构在性能上的瓶颈在哪里？/笔记.doc**

如果说你走底层的HTTP协议来进行通信，比如说你刚开始三次握手建立了一个连接，然后上传了一个文件，过了一会儿你又要上传一个文件

Spring Boot依托于一个Tomcat来启动一个Web系统，Tomcat里有工作线程300个，文件上传和下载的请求是比较耗时的，大文件的上传，要搞几分钟，甚至几十分钟都有这种可能的，这里很大的一个问题

同时有300个客户端都要上传文件的话，此时会占满这个Tomcat所有的工作线程的数量，此时就没法处理别的请求了

一个请求一个线程，没法支撑高并发，有限机器资源可能宕机

性能很差，每个请求必须把文件写入到文件存储中（之前自己研发的分布式海量小文件存储系统），才能返回响应，文件越大越耗时，时间太长会导致请求超时

一般限制为文件大小在二三十MB以内，不能太大

走普通的HTTP的请求，很可能会因为传输大文件时间耗时过长，就会导致说，请求超时，限制文件大小，比如说不能超过10MB，20MB，走普通的HTTP上传文件最大的问题就在于这里

HTTP请求超时了

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/04\_一种简单的性能优化方案：基于本地磁盘的文件缓存机制/笔记.doc**

可用性：一台机器宕机，丢失一批文件

集群部署：文件没法共享

无法进行集群伸缩

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/12\_为什么DFS用NIO自研，而云盘系统后端用Netty？/笔记.doc**

DFS，网络通信都是走NIO自研的，粘包和拆包的处理，协议的设计

发现Netty框架做的很好，但是源码写的非常复杂难懂，很难看懂Netty源码，选择技术方案，仿照Kafka网络通信架构，是很成熟的，NIO自研，仿照Kafka基于NIO自研了一套网络通信架构

云盘系统他没那么复杂，这个系统一般不那么容易出问题，也没那么大的需求要去进行扩展，所以选择了采用开源的Netty，避免为简单系统开发复杂的网络通信程序

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/01~15资料/09\_对架构的思考：如何保证大量客户端高并发传输文件？/笔记.doc**

用netty，100个线程，可能可以支撑1万个客户端，每个线程去连接100个客户端，这100个客户端发送请求过来，他依次进行处理

同时有300个线程，同时有300个客户端都在发送文件上传的请求，每个线程处理一个客户端的请求，此时又有200个客户端要发送文件上传的请求了，这个200个客户端会先跟Netty里的Accepotr线程建立连接，然后Acceptor线程会把客户端分配给Processor线程

如果Processor线程正在处理别的请求，200个客户端可以把请求发送过来，等Processor线程处理完别的请求之后，他们自己会再次多路复用轮询一下，返现有新的请求过来，就可以处理了

Tomcat，300个线程，处理300个请求，200个客户端要上传，甚至都没法跟Tomcat进行连接

Netty多路复用模式来支撑

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/18\_在代码中演示一下基于换行符的粘包拆包解决方案/笔记.doc**

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/28\_项目作业7：完成云盘客户端的开发/笔记.doc**

云盘系统，非常简单，涉及技术点，数据库表结构设计，缓存架构设计，Netty文件传输，基于ZK的协调服务，架构设计，DFS

IM系统，消息推送系统，复杂架构，核心代码还是得我带着手敲，留一些作业，DFS，留一些边缘性的东西交给你自己来做

DFS带上云盘、自己公司的文件业务，中间件系统，基础业务系统，IM系统，消息推送系统，分布式服务注册中心

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/17\_Netty中解决粘包拆包问题的核心思路是什么？/笔记.doc**

比如说客户端现在发送给服务端N条数据

“Hello World”\n

“Hello World”\n

“Hello World”\n

服务端，拆包，“Hell，下一次接收，o World”\n -> “Hello World”

服务端，粘包，“Hello World”\n“Hell

“Hello World”

“Hell

o World”\n

“Hello World”

自定义协议，请求头，定义好文件有多长，收文件的时候，就必须要读到那么长的文件次可以

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/24\_项目作业3：结合ZooKeeper设计云盘文件系统架构/笔记.doc**

在zk的/nodes目录下写一个临时节点

云盘文件系统负载均衡服务，监听zk的/nodes节点，里面的子节点有变动，就会收到一个通知

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/20\_基于自定义符号的拆包拆包解决方案演示/笔记.doc**

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/23\_项目作业2：结合亿级流量课程设计云盘业务系统缓存架构/笔记.doc**

亿级流量课程，跟jeverson要一下

Redis、缓存架构

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/26\_项目作业5：基于虚拟机部署Redis、ZK等基础系统/笔记.doc**

亿级流量课，redis、zk，虚拟机如何来部署

DFS系统，完全可以DFS，NameNode，DataNode，BackupNode，打包成一个jar包，放到linux机器上去，用java -jar命令给跑起来，用后台进程的方式跑起来，加入log4j日志，加入一些metrics统计

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/22\_项目作业1：根据云盘功能点设计数据库表/笔记.doc**

数据库表不知道怎么设计？

练习，光是搞了我们的电商系统跟着我做一遍，我的建表语句直接拿出来

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/16\_再谈TCP网络通信中的粘包和拆包问题/笔记.doc**

文件传输的话，第一个需要解决的，就是小文件的粘包和拆包的处理

TCP

客户端和服务端

客户端发送一个10MB的小文件出去，但是底层使用了2个包来传输，服务端先收到一个5MB的半包文件，又收到5MB的半包文件

客户端发送两个10KB的小包，在底层合并为了一个大的网络包发送过去，服务端收到的是一个大的20KB的大包

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/27\_项目作业6：完成云盘业务系统、文件系统和协调服务的开发/笔记.doc**

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/19\_运行示例代码测试一下基于换行符的粘包拆包解决方案/笔记.doc**

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/25\_项目作业4：云盘客户端的容错机制设计/笔记.doc**

很多东西，都跟我们之前的dfs系统是类似的

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/16~30资料/21\_基于自定义固定长度的拆包粘包解决方案介绍/笔记.doc**

FixedLengthFrameDecoder(30)，字节长度，截取出来作为一条数据

发送的每个数据长度都是一样的，此时才可以用这样的方案

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/109\_基于SSL的消息加密传输以及自己试验Netty SSL的小作业/笔记.docx**

《109\_基于SSL的消息加密传输以及自己试验Netty SSL的小作业》

IM系统，一般来说在进行消息通信的时候最好是进行加密传输，SSL，发送消息的时候进行加密，在消息在传输过程中是加密的状态，即使被人截获了也是不知道的，在服务器端进行解密

Netty直接是支持SSL加密传输的

自己去查阅一些资料，Netty SSL，一大堆的Demo就出来了，实验一下基于Netty的SSL加密传输通信

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/122\_如何将IM课程中学习到的知识彻底消化为自己的东西？/笔记.docx**

《122\_如何将IM课程中学习到的知识彻底消化为自己的东西？》

三周的IM架构方案讲解的课程，一步一步把架构完善，一步一步把对应的业务功能，自己撸出来，一步一步结合我之前的亿级流量、面试突击、分布式事务，缓存架构、分库分表、可靠消息最终 一致性方案

把IM系统的架构一步一步撸出来，各种业务功能和技术方案一步一步撸出来

最终你会做出来一个估值几亿美金到几十亿美金的独角兽互联网公司或者是中小型公司需要使用的IM系统的，生产级、企业级的架构，你就全都撸出来了

对很多同学，如果明年要出去找工作，基于这样一套企业级的、生产级的IM系统，是绝对没有问题的

DFS系统、IM系统，11中旬，自研微服务注册中心系统，Dubbo源码，自研服务治理平台

DFS + IM + 服务注册中心（用来替代Spring Cloud里的Eureka），生产级、企业级的架构，大量的技术方案，底层的技术细节，自研分布式系统，架构班目前迭代到现在为止的分布式电商架构、亿级流量

30k~35k，你是有这样的技术实力

20多k~30多k，其实都是可以的，项目是非常有技术含金量，硬碰硬，近一些知名的大公司都是可以的

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/121\_如何实现收发消息后自动更新会话列表顺序的功能？/笔记.docx**

《121\_如何实现收发消息后自动更新会话列表顺序的功能？》

你只要跟某个用户打开一个新的会话，对话框，此时就需要发送一个请求到后台，会话列表服务，此时就需要在Redis里维护石杉的一个会话列表，在里面加入一个新的会话

如果说此时，会话里，跟李四的会话排在第一位

但是此时，你跟张三发送了一条消息，或者是张三发送了一条消息给你，此时你就应该改变session顺序

Hash + SortedSet作为一个数据结构

Hash，在里面我们是存放所有石杉打开的会话

{

“session\_张三\_id”: “SortedSet里的一个索引，0”,

“session\_李四\_id”: “SortedSet里的一个索引，1”

}

[100 ->“session\_张三”, 101 ->“session\_李四”]

优化一下，客户端不需要在打开新的会话的时候发送请求给你，你就只要在后台收听消息的收发就可以了，如果发现有你发送新消息，或者人家发送消息给你，此时看一下你的会话列表里，有没有跟这个人的session

如果没有，自动加入一个session进去，更新他们的顺序就可以了

如果说你在另外一个设备上登录了软件，此时要加载已有的会话出来，直接通过会话列表服务去加载就可以了

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/114\_红包功能优化：单条群红包消息写入Kafka以及群成员Session聚合/笔记.docx**

《114\_红包功能优化：单条群红包消息写入Kafka以及群成员Session聚合》

对于群红包消息，在红包服务里进行特殊处理，不要对群里500个人生成500条消息反向推送，仅仅在这里就是把一条群红包消息推送到kafka里去就可以了

对于消息投递系统，他需要根据群id一次性获取出来群里500个人的session数据

群成员session聚合，对于我们的session管理服务，每次有一个人连接之后完成认证，都会在redis中写入他的session，粒度是一个用户一个session，其实是这样子，会话管理服务，群成员session聚合

查询一下用户属于哪些群，对于每个group都得在redis里搞一个群session

key: group\_session\_#{groupId}

value: 就是群里每个成员的session都在这里，[{}, {}, {}]

session，几十个字节，500个成员，几万个字节 -> 几十KB，100KB

直接根据groupId到redis里查一次，就直接查出来这个群里所有人的session，此时根据每个群里在线的人的sessino，定位到TCP接入系统，分发红包消息过去，由接入系统反向推送红包消息给那个人就可以了

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/110\_如果要在IM系统中实现抢红包功能有哪些技术难点？/笔记.docx**

《110\_如果要在IM系统中实现抢红包功能有哪些技术难点？》

角色的拆分

IM系统比较常见的一个功能就是发红包，c2c的红包，IM系统不常见，群里面发红包和抢红包可能是会存在的

发红包这个事情是涉及到交易和转账的，所以肯定不是简单的耦合在c2c或者g2g服务里的，所以红包要单独拉出来，另外一个人收到了红包消息，点击收红包，此时必然会发送请求到后台找红包服务

DFS系统，IM系统，支付系统

红包服务 -> 支付系统的接口，进行一个转账操作，用户A的某个账户里的钱进入用户B的账户里去，涉及到一个交易转账的过程，需要你跟支付系统进行交互。微信红包，给别人发红包，选择这个钱是从哪儿过去的？

发红包的时候，红包服务就需要做一次交易转账，调用支付系统，把用户银行卡账号里的钱转入到你公司的对公银行卡账号里去，后台还得有定时线程，如果一个红包超过24小时没人领取，此时就要从你公司的账号里原路退还给用户

走支付系统进行交易转账

红包的功能放在IM系统里来做，要配合支付系统来搞的

群里抢红包的功能，微信群里发红包，指定红包的个数，10个红包，一共是150块钱，10个红包里会用一个随机算法去分配150到10个红包里去，会生成10个红包消息，每个红包是多少钱

围绕红包功能设计一堆的表

此时比如群里有100个人，你还得生成100条红包消息反向推送给100个人，让这100个人都立马看到有一个群红包发出来了，然后他们才能来抢红包，红包服务还得有一个抢红包的业务逻辑

原子性，最终只能是10个红包被抢走，100个人里最多只能有10个人抢走红包，每个人会抢到一个红包，走一个支付系统进行交易转账

对于普通的群消息，反正无非就是一个看而已，如果有一个人在500人群里发了普通的文字消息，此时你可以慢慢反向推送都无所谓，哪怕是对消息做一些排队都可以，500个人看到别人发的文字消息，早点晚点是无所谓的

抢红包，500个人的群，抢红包，谁先看到红包消息就会先抢，谁晚点看到红包消息就会抢不到了，公平的让这500个人保证他们看到红包消息的顺序，保证大家抢红包的公平性

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/113\_红包功能优化：独立为红包消息部署一套消息投递系统/笔记.docx**

《113\_红包功能优化：独立为红包消息部署一套消息投递系统》

比如说现在有1万条普通消息积压在kafka里，还有500条红包消息积压在1万条普通消息的后面，此时对于消息投递系统而言，他必须先发送完1万条普通消息，然后才能发送500条红包消息

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/112\_分析一下最基本的抢红包功能在生产环境中存在哪些问题？/笔记.docx**

《112\_分析一下最基本的抢红包功能在生产环境中存在哪些问题？》

红包的延时还是太大了

比如有一个群是500个人的群，其中有10个人在一起面对面的在聚餐，有一个人在群里发了一个红包之后，也许几秒过后另外9个人才可以在群里看到这个红包消息，红包这种低延迟的消息，速度还是太慢了

一条群红包要变成500个消息写入redis cluster，耗时很大；500条消息会依次写入kafka中，耗时很大；消息投递系统，要在Redis里检索500次用户session状态，然后转发给接入系统，耗时很大；

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/117\_一个完整红包功能的架构设计和业务流程总结（续）/笔记.docx**

《117\_一个完整红包功能的架构设计和业务流程总结（续）》

群红包

得做一个总金额到指定数量的群红包的金额随机拆分，把10个红包插入到数据库中去，另外做一个发送用户到公司账户的转账，投递一条群红包消息到Kafka中去

红包独立部署的离线消息服务，会马力全开，为群500个人生成的离线消息，利用几十个甚至上百个线程同时并发写入Redis中，几十ms就可以了

调用公司自己的支付系统 -> 第三方的支付系统（微信、支付宝、银行），这个过程就慢了，快则几百毫秒，慢则几秒，这个很不稳定

对于调用公司自己的支付系统进行交易转账这个事儿，完全可以采用分布式事务里面的可靠消息最终一致性方案，最终一定要调用支付系统成功把交易转账完成，我的另外一个好朋友，狸猫技术窝，《从0开始带你成为消息中间件实战高手》

RocketMQ这个技术

基于RocketMQ实现一套可靠消息最终一致性的分布式事务方案

红包独立部署的消息投递系统，走群Session聚合查询，一次Redis调用，群里所有在线的人的session，做消息投递

群里500个人，如果说收到了群红包消息之后，删除他自己的群红包离线消息，抢红包，走红包服务的抢红包的逻辑，基于Redis来实现，你完全可以针对每个群红包在Redis里写一条数据，他的值是10

在抢红包的时候，就是他的逻辑就是堆Redis里的这个数值进行原子性的扣减，如果红包数量还大于0，此时就可以继续允许别人来抢红包

如果有人抢到了红包，此时就是在数据库里更新那个小红包的状态是已领取，发送消息到MQ进行红包的交易转账，红包数据库里应该是有两张表

对于一个群红包而言，他是一条记录 -> 很多小红包，小红包表里有多条记录

其他的红包表，红包领取表，对于群里的每个小红包，需要有一个小红包领取关系表，记录下来是哪个用户领取了哪个小红包，领取的小红包是多少钱

定时扫描服务，扫描群红包里的每个小红包的状态，如果超过24小时有的小红包还没有人领取，此时就退款剩余没领的钱给发送红包的用户就可以了

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/115\_红包功能优化：独立部署红包离线消息服务多线程个并发写Redis/笔记.docx**

《115\_红包功能优化：独立部署红包离线消息服务多线程并发写Redis》

专门针对红包独立部署，开启多线程写入redis，一个群消息，对应一个500人的群，500条消息，500次写入redis，单个线程500次写入redis很慢，开启多个线程，加足马力，全力写入群红包消息

4核8G的机器，100个线程，500条群红包消息分配给100个线程，每个线程就写5条，25ms，100个线程额同时写500个消息，仅仅花费25ms就可以写完500条消息

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/123\_学完截止目前的IM课程后还缺了哪些东西以及最后一周IM课程预告/笔记.docx**

《123\_学完截止目前的IM课程后还缺了哪些东西以及最后一周IM课程预告》

每个人一打开自己的APP，读一下自己当前未读消息的总数，如何让大图片、大视频、大音频的上传更加的有效率、如何加速我们的图片的读取、如何高可用（熔断、限流、降级，Hystrix）、如何把缓存做成多级缓存的架构、网路传输的安全性、分布式锁也要融入到这个系统架构里去、监控、运维、扩容缩容

希望大家一定要按照我说的，IM系统是绝对企业级、生产级

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/111\_如何在IM系统中实现一个基本的抢红包功能？/笔记.docx**

《111\_如何在IM系统中实现一个基本的抢红包功能？》

大体的抢红包的公平性

群里有500个人，都在群信息表里，每次如果你查出来群里的人是500个人，从数据库查询出来就是有顺序的

张三

李四

王二

麻子

如果每次你都根据这个顺序生成群红包消息，然后按照这个顺序推送出去，每次都是群里的张三先看到群红包，李四，王二，麻子，10个红包，群里有10个人抢到红包，公平就是不公平了

用Collections.shuffle()，对这500个人做一个随机洗牌，打乱顺序，根据打乱的顺序去生成群红包消息，推送出去

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/105\_JWT是个什么东西？他在系统安全认证中有什么作用？/笔记.docx**

《105\_JWT是个什么东西？他在系统安全认证中有什么作用？》

对于IM系统的安全认证这块，基于SSO单点登录系统进行token认证，性能不是太好。JSON Web Token，开放式的Token的标准。授权认证，你的用户如果一旦说登录了之后，就可以授予一个JWT Token，后续直接带着这个Token就可以去访问所有被授权的URL路径，资源，服务

JWT本身也是可以用在单点登录这一块的

数据传输，基于JWT可以让你传输的数据进行签名，signature，密码或者私钥/公钥进行签名，就可以做到类似于加密数据传输这样的效果

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/119\_如何在IM系统中加入类似钉钉的消息已读功能？/笔记.docx**

《119\_如何在IM系统中加入类似钉钉的消息已读功能？》

在群里发送一条消息，会显示当前群里多少人未读，而且你点开这个消息会看到群里哪些人没读这条消息

群里发送消息，会显示多少人未读，采取推？大量推送会搞垮客户端，卡死。采取拉？间隔太长数据不准确，间隔太短，会很耗电

推拉结合？

打开客户端拉取一次未读人数，你的IM系统需要在后台建立一个表，消息未读人数表，同时发送请求到服务端订阅这个未读人数通知，有变化就反向推送，如果客户端退出就取消这个订阅，但是一样是会存在大量推消息造成的问题

群聊消息，500人，刚开始历史消息服务插入一条群聊消息，字段，未读人数是500，不断的扣减群聊消息的未读人数

还是采取拉

比如你每次打开客户端第一次看到这条消息，拉取一次，如果长时间打开客户端停留在这个消息上，可以自动降低频率，比如一开始2秒拉取一次，5秒拉取一次，或者10秒拉取一次，避免耗电

我们在客户端里会不停的滑动去看一些消息，单聊消息还是群聊消息

客户端需要控制一下，如果你滑动到某个单聊消息，此时他的状态是未读，此时就发起一次请求去拉取消息的状态，群聊消息也是一样，你滑动的时候滑动到哪儿去加载哪些消息的状态

如果你停留在某个消息上，降低拉取消息状态的频率

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/118\_类似钉钉的消息已读功能是个什么东西？/笔记.docx**

《118\_类似钉钉的消息已读功能是个什么东西？》

石杉发送了一个消息给张三，我的对话框里会显示出来一条我发送出去的这个消息，这个东西对于QQ和微信，我们是不知道的

钉钉，张三如果还没收到消息，没读这个消息，此时显示的状态是未读

如果张三收到了这个消息，读了这条消息，此时我的对话框里会显示出来这个消息的状态是已读

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/106\_JWT的数据结构：header、payload和signature/笔记.docx**

《106\_JWT的数据结构：header、payload和signature》

header.payload.signature

header：token类型，加密算法，比如说{“typ”: “JWT”, “alg”: “HS256”}，Base64Url进行编码

payload：claim，包含了用户状态和一些元数据

signature：对header和payload进行数据签名，用算法生成hash，这样可以确保数据不会篡改，啥意思？signature(header + payload) = 算法生成的hash。如果你收到一个JWT token，可以利用同样的算法对header和payload进行hash，跟signature比对，如果一样就没篡改，否则就篡改了

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/120\_什么情况下需要在服务端保存客户端的会话列表？/笔记.docx**

《120\_什么情况下需要在服务端保存客户端的会话列表？》

通信软件，他需要有一套自己的后台系统，账号系统，联系人系统，加好友，发起会话，客户端里有一个会话列表，你跟哪些人发起过会话，如果你的某个会话发送了消息或者是收到了消息，都会自动把有新消息收发的会话放到你会话列表最上面去

会话列表可以放在你的客户端软件自己本地存储的

比如说你现在有两部手机，一个是iphone，一个是小米，平时你一般是用iphone登录IM软件，某一天突然你在小米上去登录聊天软件，此时你的小米上是没有你的iphone上本地保存的会话列表

会话列表是空的

你还需要在服务端存储你的会话列表

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/108\_一个作业：自己查阅资料动手实验SSO和JWT的认证流程/笔记.docx**

《108\_一个作业：自己查阅资料动手实验SSO和JWT的认证流程》

系统的安全性架构，专题课程

玩玩儿SSO单点登录，demo；JWT的demo；demo -> 请求SSO进行单点登录，获取sso token -> 请求JWT server，获取jwttoken -> 如何在服务端单纯基于约定的算法对jwttoken完成认证

在IM系统里，这块东西，你出去面试讲这个系统，生产级的认证流程

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/104\_基于SSO单点登录系统进行认证的缺陷分析/笔记.docx**

《104\_基于SSO单点登录系统进行认证的缺陷分析》

反复执行认证的过程会对SSO单点登录系统造成过大的压力，反复重连，每次重连都要请求SSO单点登录系统，这样会导致每次重连的性能都很差，要耗费很多的时间进行SSO单点登录系统的认证

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/116\_一个完整红包功能的架构设计和业务流程总结/笔记.docx**

《116\_一个完整红包功能的架构设计和业务流程总结》

c2c的红包消息

调用支付系统，做一次发送用户的账户到公司账户的转账，就可以保证接收用户随时可以领取红包里的金额了，微信发红包，你一旦发红包出去，人家不管有没有领取，此时都会通知你说已经完成了一次转账交易

在数据库里插入一条红包消息，状态是未领取

投递红包消息到Kafka

红包功能独立部署的离线消息服务，写入离线消息到redis里去

红包功能独立部署的消息投递系统，将红包消息投递出去给接收用户，如果用户接收到了红包消息之后，ACK传回来，红包服务删除Redis里的离线消息

如果接收用户点击领取红包，红包服务调用支付系统进行交易转账，公司账户转账到接收人账户里去，数据库里的红包状态更新为已领取

红包服务后台线程定时扫描数据库里的红包消息，如果超过24小时未领取，则直接调用支付系统进行原路退还，红包状态更新为已过期，后续不能再领取红包了

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/104~123资料/107\_基于SSO单点登录和JWT共同完成的优化后的认证流程/笔记.docx**

《107\_基于SSO单点登录和JWT共同完成的优化后的认证流程》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/020~029资料/029\_初步体验一下Protobuf序列化和反序列化的效果/笔记.docx**

《029\_初步体验一下Protobuf序列化和反序列化的效果》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/020~029资料/020\_本周核心内容：接入层与分发层通信、自定义协议与序列化，等等/笔记.docx**

《020\_本周核心内容：接入层与分发层通信、自定义协议与序列化，等等》

上一周的课程，APP SDK <-> 接入层之间的双向通信都已经打通了

建立物理连接，发起认证请求，接入层会缓存跟客户端的长连接，客户端也会缓存跟接入层的长连接，客户端随时可以通过长连接向接入层发送请求，接入层也随时可以反向主动发送消息给客户端

本周几块内容：接入系统，网关系统一般用在微服务架构里，需要跟分发系统进行交互；基于TCP的自定义协议，序列化机制；核心功能搞定，认证功能，session管理（接入层本地Session，基于Redis的分布式Session管理），登出（logout），踢出（kickout）

下周的内容：Kafka（大数据架构的课程里讲解，源码，实战）；单聊，群聊，离线消息，历史消息，数据库表结构设计，请求/响应的协议设计

十一

十月份，我们完成最终版本的IM系统：完善复杂的大型IM架构、数据库分库分表、timeline消息存储模型、消息有序性/时效性、安全性（token认证）/消息加密存储

企业级的IM系统，就搞定了，二次开发，直接可以应对企业级的IM通信的需求的

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/020~029资料/026\_一次客户端认证请求的全流程分析/笔记.docx**

《026\_一次客户端认证请求的全流程分析》

我们这周的内容需要把认证请求的全流程写完的，logout，kickout

自定义协议 + Protobuf序列化

整体测试跑通目前做好的全流程，只要都搞定了，基本上就差不多了

Redis Cluster暂时我们这周不部署，下周跟Kafka一块儿部署，MySQL数据库，单聊，群聊，离线消息，核心功能都做了

10月底，这个项目就全部搞定，比较完整的企业级IM系统的项目经历，分库分表，安全认证，timeline模型，有一定难度的技术都会有，Redis大量使用，缓存架构，结合亿级流量那个课程，把一些东西都可以融入在这里

业务系统上没什么出彩的东西，就完全可以在明年找工作的时候，用我们的今年的基础架构的一些项目去说

11月份，可以搞一个Push系统，要简单很多，Netty的一些技术

12月份~1月份，有一个重量级的项目，就是自研升级版本的分布式微服务注册中心系统，难度比较高，很有含金量，Netty都搞定了，所以底层的网络通信都可以搞定了，在这个过程中，就会顺带把ZooKeeper

有可能是放在大数据架构里去讲解的

ZooKeeper这个技术，可能会在10月份~12月份之间，在大数据架构里开一个课来深入讲解

Java架构而言，1月底，过年期间，年前年后，基于ZooKeeper做一点点开发，分布式海量小文件存储系统，Master改成HA，基于ZK来做；分布式微服务注册中心、IM系统、Push系统，都会用到ZooKeeper做一点改造

2019年辛苦一年，基础架构彻底搞定

基础架构的一些经验可以拿出来出去说了，里面很多技术都是很有价值的

2020年年初开始，就得做复杂的业务系统架构了，微服务、海量数据、高性能、高并发、高可用、安全性、中台架构 -> 大量的中间件的源码研究，RocketMQ、Sharding-Sphere、MyCat、Redis、Elasticsearch

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/020~029资料/028\_基于Protobuf生成Java类以及初步体验序列化的效果/笔记.docx**

《028\_基于Protobuf生成Java类以及初步体验序列化的效果》

第一个文件：AuthenticateRequest.ptoto

package im;

option java\_package = “com.zhss.im.protocol”;

option java\_outer\_classname = “AuthenticateRequestProto”;

message AuthenticateRequest {

string uid = 1;

string token = 2;

int64 timestamp = 3;

}

第二个文件：AuthenticateResponse.ptoto

package im;

option java\_package = “com.zhss.im.protocol”;

option java\_outer\_classname = “AuthenticateResponseProto”;

message AuthenticateResponse {

int32 status = 1;

int32 errorCode = 2;

string errorMessage = 3;

}

官网下载Protobuf：https://github.com/protocolbuffers/protobuf/releases/tag/v3.9.1

下载：protobuf-java-3.9.1.tar.gz

windows命令行上就可以用了，mac是类unix的操作系统

安装mac上的brew：/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

安装一些基础类库：brew install autoconf automake libtool

tar -zxf protobuf-3.9.1.tar.gz

cd protobuf-3.9.1

./autogen.sh

./configure

make

make check

sudo make install

protoc -version

protoc –java\_out=im/src im/proto/AuthenticateRequest.proto

新建工程，im-protocol，引入依赖，protobuf-java的依赖，拷贝生成的类进去

syntax = "proto3";

option java\_outer\_classname = "AuthenticateRequestProto";

message AuthenticateRequest {

string uid = 1;

string token = 2;

int64 timestamp = 3;

}

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/020~029资料/024\_初步实现接入系统与多个分发系统实例建立连接的逻辑/笔记.docx**

《024\_初步实现接入系统与多个分发系统实例建立连接的逻辑》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/020~029资料/025\_整体梳理一下系统架构以及认证请求流程/笔记.docx**

《025\_整体梳理一下系统架构以及认证请求流程》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/020~029资料/023\_继续搭建分发系统的Netty服务端通信框架/笔记.docx**

《023\_继续搭建分发系统的Netty服务端通信框架》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/020~029资料/021\_接入系统与分发系统如何进行双向网络通信？/笔记.docx**

《021\_接入系统与分发系统如何进行双向网络通信？》

接入系统收到一个请求，要转交给分发系统，此时走一个普通的Spring Cloud做一个微服务之间的RPC调用，也是可以的。分发系统提供HTTP接口，接入系统直接RPC调用他的接口就可以了

TCP接入系统，假设每台机器可以支撑10万个客户端的长连接；此时如果假设有100万个客户端同时要建立长连接呢？此时你就需要把TCP接入系统部署10台机器

某个客户端他只是跟1台接入系统的机器进行了连接的

分发系统需要知道推送给1个客户端的请求，那个客户端跟哪个接入系统建立了长连接呢？

分发系统通过Spring Cloud去发送请求给接入系统，是不靠谱的，负载均衡，你是不能控制分发系统把请求推送给哪台接入系统的机器的

TCP接入系统跟请求分发系统，要建立长连接，每台接入系统可以连接到所有的分发系统，每台分发系统也可以连接到所有的接入系统

接入系统要把请求发送给分发系统，此时可以随机挑选一台分发系统出来，请求发送过去就可以了；分发系统把请求发送给接入系统，此时需要根据你在Redis中存储的Session信息，根据uid（用户id），去Redis中查找到Session

通过Session就知道这个uid的长连接是跟哪台接入系统的机器建立起来的，此时把请求推送给那台接入系统的机器就可以了，接入系统再根据uid找到自己机器上存储的本地的Session信息，知道uid对应的长连接是哪个SocketChannel

反向推送消息过去就可以了

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/020~029资料/027\_认证请求的协议设计以及Protobuf入门/笔记.docx**

《027\_认证请求的协议设计以及Protobuf入门》

序列化，在你的一个系统里，有一块数据，可以是一个字符串，更多的时候不能用字符串，是用自定义的Java对象，就是如果你要把这个Java对象通过Netty传输给其他的系统的话，此时如何将Java对象搞成二进制字节数组呢？

把二进制字节流或者字节数组转换回Java对象，反序列化

协议如何来制定，认证请求包含哪些字段，认证请求的响应包含哪些字段

协议对应的请求和响应都是Java对象

如何将Java对象进行序列化和反序列化，在网络上传输

我们就走二进制字节数组，在里面我们会放入很多的数据，自定义协议设计，协议头，协议体，协议头

headLength，4 byte，消息头长度，固定20个字节

clientVersion，4 byte，客户端SDK版本号

requestType, 4 byte，请求类型

sequence，4 byte，请求序号

bodyLength，4 byte，请求体长度

Protobuf进行序列化，就是对请求体是复杂的Java对象，Protobuf序列化成数组就可以了

第一个文件：AuthenticateRequest.ptoto

package im;

option java\_package = “com.zhss.im.protocol”;

option java\_outer\_classname = “AuthenticateRequestProto”;

message AuthenticateRequest {

string uid = 1;

string token = 2;

int64 timestamp = 3;

}

第二个文件：AuthenticateResponse.ptoto

package im;

option java\_package = “com.zhss.im.protocol”;

option java\_outer\_classname = “AuthenticateResponseProto”;

message AuthenticateResponse {

int32 status = 1;

int32 errorCode = 2;

string errorMessage = 3;

}

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/020~029资料/022\_初步搭建分发系统的网络通信框架/笔记.docx**

《022\_初步搭建分发系统的网络通信框架》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/038\_完成分发系统对认证请求的整体处理逻辑/笔记.docx**

《038\_完成分发系统对认证请求的整体处理逻辑》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/043\_在认证请求全流程中加入详细的调试信息/笔记.docx**

《043\_在认证请求全流程中加入详细的调试信息》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/035\_为IM系统开发通用工程来提供公用的代码组件/笔记.docx**

《035\_为IM系统开发通用工程来提供公用的代码组件》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/034\_在TCP接入系统中实现响应消息封装逻辑/笔记.docx**

《034\_在TCP接入系统中实现响应消息封装逻辑》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/041\_重构公共工程里的请求和响应抽取基础父类/笔记.docx**

《041\_重构公共工程里的请求和响应抽取基础父类》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/040\_完成接入系统将响应路由到对应的APP长连接并返回过去/笔记.docx**

《040\_完成接入系统将响应路由到对应的APP长连接并返回过去》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/033\_在TCP接入系统中开发认证请求的处理逻辑/笔记.docx**

《033\_在TCP接入系统中开发认证请求的处理逻辑》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/039\_完成分发系统到接入系统再到APP SDK的响应返回/笔记.docx**

《039\_完成分发系统到接入系统再到APP SDK的响应返回》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/030\_本周内容介绍以及IM系统后续课程进度安排的通知/笔记.docx**

《030\_本周内容介绍以及IM系统后续课程进度安排的通知》

出了点意外，抓紧时间，序列化，认证请求，搞通的，Mac电脑上用Protobuf特别的麻烦，2.5版本都很简单的。你其实还可以定义服务和接口，人家会自己给你生成的代码，就会在底层进行网络通信和调用

RPC框架，Thrift，他就可以实现类似这样的效果，类似于Protobuf的语法定义服务和接口，类，生成一堆类，直接用这些类就可以实现网络通信和RPC调用

IM系统，最核心的主要是基于Netty来实战，实现IM核心功能，给大家说一说其他的架构的设计，缓存架构，分库分表，Kafka，亿级流量那一套架构就可以实现，Redis技术的讲解，分库分表，面试突击第一季，Kafka

网络通信架构、缓存架构、分库分表架构、Kafka消息架构

网络通信架构 -> IM系统架构搭建出来 -> 实现核心功能 -> timeline消息模型 / 消息顺序性 / 消息时效性 / 高阶功能 / 安全认证 -> 缓存架构 / 分库分表架构 / Kafka消息架构（结合以前的课程）

就是一个非常完整的企业级的、工业级的消息系统的架构了

headLength，4 byte，消息头长度，固定20个字节

clientVersion，4 byte，客户端SDK版本号

requestType, 4 byte，请求类型

sequence，4 byte，请求序号

bodyLength，4 byte，请求体长度

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/042\_重新梳理认证请求全流程以及采用最新公共消息类/笔记.docx**

《042\_重新梳理认证请求全流程以及采用最新公共消息类》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/032\_在TCP接入系统中解析收到的认证请求/笔记.docx**

《032\_在TCP接入系统中解析收到的认证请求》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/036\_实现一个通用的响应解析业务逻辑/笔记.docx**

《036\_实现一个通用的响应解析业务逻辑》

稍微听一下脚步，做一个总结和梳理，看看目前为止的整个流程是怎么样的

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/031\_在APP SDK中采用自定义协议和Protobuf序列化封装认证请求/笔记.docx**

《031\_在APP SDK中采用自定义协议和Protobuf序列化封装认证请求》

有这么几个头

headerLength，4 byte，消息头长度，固定20个字节

appSdkVersion，4 byte，客户端SDK版本号

requestType, 4 byte，请求类型

sequence，4 byte，请求序号

bodyLength，4 byte，请求体长度

还有消息体，解析出来了一个请求，接下来就根据请求的类型进行判断，一点一点来进行处理

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/044\_完成认证请求从SDK APP到接入系统到分发系统的全链路测试/笔记.docx**

《044\_完成认证请求从SDK APP到接入系统到分发系统的全链路测试》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/030~044资料/037\_利用已经开发好的基础代码实现请求转发给分发系统/笔记.docx**

《037\_利用已经开发好的基础代码实现请求转发给分发系统》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/014\_初步做一个Token认证伪代码流程以及长连接缓存机制/笔记.docx**

《014\_初步做一个Token认证伪代码流程以及长连接缓存机制》

作为一个APP而言，如果说这个用户他要打开一个APP上的“联系客服”的按钮和功能的话，首先APP得保证用户先得登录，系统通用的SSO单点登录系统，所有的系统只要登录一次，后面所有的其他的系统都用可以走一个已经登录过的这个用户就可以了

userId：代表的是这个用户的唯一标识

token：SSO单点登录的机制去讲解，令牌，代表这个用户他已经登录了

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/016\_初步实现APP SDK向TCP接入系统发送消息的逻辑/笔记.docx**

《016\_初步实现APP SDK向TCP接入系统发送消息的逻辑》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/011\_如何基于Netty进行双向通信？/笔记.docx**

《011\_如何基于Netty进行双向通信？》

之前讲解了一些Netty的Demo级别的代码，是不足以去开发一个项目的，这也是为什么，我们必须要做项目来熟练的使用一个技术

如果是支持社交的功能，多个使用APP的C端用户，他们可能会建立一个群聊，在群聊里，他们可以进行IM即时通讯

Netty服务端可以接收客户端发送过来的消息，也可以反向推送消息回客户端去

就是我们的Netty服务端每次都是在接收到客户端的请求之后应答一个响应回去，他没有办法主动推送一个消息给客户端去

网络通信的双向通信的架构给搞出来

底层的网络连接通过Netty建立好了之后，其实应该是客户端发送一个token过来，进行认证，服务端对token认证过后，才能认为连接建立完毕的，否则token认证失败的话，此时就应该关闭这个连接

一旦token认证成功了之后，是不是就可以在服务端把这个客户端的连接给缓存起来

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/010\_基于Netty初步搭建TCP接入系统的网络通信框架/笔记.docx**

《010\_基于Netty初步搭建TCP接入系统的网络通信框架》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/015\_初步为TCP接入系统开发一个反向推送程序/笔记.docx**

《015\_初步为TCP接入系统开发一个反向推送程序》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/013\_APP SDK中如何在连接建立之后保存SocketChannel？/笔记.docx**

《013\_APP SDK中如何在连接建立之后保存SocketChannel？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/008\_从哪里开始入手？客户端SDK、接入层以编码解码/笔记.docx**

《008\_从哪里开始入手？客户端SDK、接入层以编码解码》

IM即时通讯系统，第八讲，即时通讯系统，消息Push系统，两个系统，是实战Netty和一系列的底层技术最好的两个项目

老的Web系统：SSH + Tomcat + MySQL + Redis/MQ/NoSQL

新的Web系统：Spring Boot + SSM + Spring Cloud/Dubbo + MySQL + Redis/MQ/NoSQL

HTTP协议对外提供一系列的HTTP接口，一般来说是开发各种各样的业务系统，电商系统，O2O系统，教育系统，医疗系统，政务系统

集合（内存） + 并发 + 网络 + IO（磁盘）

中间件系统：分布式海量小文件存储系统，分布式服务注册中心，存储、微服务、消息、缓存，等等，诸如此类的一些系统

基础系统：跟业务无关，基础性的一些系统，支撑上层的各种业务，跟中间件系统有点类似，IM即时通讯系统，消息Push系统，爬虫系统，地图系统，搜索系统

基础架构系统，业务无关，支撑所有的业务，很多中小型公司可能不会花费很多时间，基础架构团队，维护一些开源项目/定制开发，也可能会自研一些基础架构系统

2019年一年，全部都是focus在底层技术 + 基础架构这块的

即时通讯，IM，Instance Message，即时消息系统

客户端 + 接入层 + 编码解码（序列化协议）

客户端如何跟服务端建立长连接进行双向通信，接入层作为IM系统的网络层，负责管理网络连接，维持大量的长连接，客户端和服务端进行通信的时候使用什么样的序列化机制去把复杂的对象转换为二进制字节流

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/009\_为什么接入层的TCP和WebSocket要拆分为两套系统？/笔记.docx**

《009\_为什么接入层的TCP和WebSocket要拆分为两套系统？》

电脑切换成mac之后，主要是基于IntelliJ IDEA去开发了，因为以前的那个二手笔记本电脑，比较老了，当时用的是Eclipse，但是实际上现在就是说大部分的工程师，一般用的IDE都是IntelliJ IDEA了

为什么tcp接入系统和websocket接入系统要分开开发和部署呢？

TCP接入系统主要是对接大量的C端用户，比如说你的电商APP每天会活跃百万用户，每天都有几十万的用户会来跟你的TCP接入系统进行连接，跟客服进行沟通

WebSocket接入系统主要是对接公司里的几十个，或者几百个客服而已，主要是客服会通过浏览器打开一个网页，网页通过HTML5的WebSocket API，跟你的WebSocket接入系统建立连接

TCP接入系统面对的百万级甚至千万级的C端用户，负载很高的，所以部署大量的机器

WebSocket接入系统面对的可能仅仅是几百个公司内部的客服而已，负载很低，所以可能就不需要部署大量的机器

使用的协议和技术也不太一样，部署的机器需求也不太一样

TCP从网络通信的一些代码，通信协议，序列化协议，都是跟WebSocket是不一样的。WebSocket是一种独立的网络通信协议，序列化协议，可能都有区别的

TCP接入系统和WebSocket接入系统，分别给开发出来

分别去开发对应的客户端的SDK，一种是给APP用的SDK，是会基于TCP协议去接入和通信；一种是给网页用的SDK，是会基于WebSocket去接入和通信

序列化，复杂数据如何序列化成字节流

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/012\_基于Netty初步搭建APP SDK的通信框架/笔记.docx**

《012\_基于Netty初步搭建APP SDK的通信框架》

APP SDK，我们主要是针对Android的，本来就是基于Java语言的，IOS的话，这个，确实我对IOS以及对应的开发语言，也不是太熟悉，如果你真要是给IOS开发SDK的话，那你必须得熟悉IOS上的一些开发语言和对应的网络通信框架

熟悉IOS开发的同学来帮你进行封装，你仅仅提供TCP的长连接的能力，具体如何来连接，交给IOS开发组自己来封装一个网络通信的SDK出来

Netty客户端，就可以封装出来Android SDK

添加一些网络交互的一些流程，包括认证，SocketChannel的缓存，如何进行通信

要实现网络通信的双向性，一旦连接建立完毕之后，客户端随时可以发送请求给服务端；服务端也可以随时发送消息给客户端

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/018\_初步完成APP SDK与TCP接入系统的网络通信联调/笔记.docx**

《018\_初步完成APP SDK与TCP接入系统的网络通信联调》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/019\_测试时发现一个Netty粘包的问题/笔记.docx**

《019\_测试时发现一个Netty粘包的问题》

我们感觉上发送了两条消息过去，结果只收到了一条认证的消息

网络通信框架搭建起来了，TCP去讲解，WebSocket这块留给大家自己做作业，非常简单的

WebSocket这块测试不太好测，还需要写前端页面的JS代码，没什么难度，我给留一个作业，这周自己开发一个WebSocket接入程序，就可以测试

TCP这块来讲解，WebSocket，很耗费时间

客户端可以跟服务端建立长连接，Netty企业级的开发，可以走用户认证的请求，长连接缓存，包括消息发送过去，消息的反向推送，连接断开的一个处理，粘包的问题

作业：把WebSocket接入系统和客户单开发出来，测试，维护长连接，要可以双向发送消息，JS代码，基于HTML5提供的WebSocket的API去开发

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/008~019资料/017\_开发APP SDK的测试程序以及加入调试信息打印/笔记.docx**

《017\_开发APP SDK的测试程序以及加入调试信息打印》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/045~055资料/046\_测试一下基于Jedis连接到Redis进行操作/笔记.docx**

《046\_测试一下基于Jedis连接到Redis进行操作》

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

<version>3.1.0</version>

</dependency>

关注在IM系统本身的实战基于Netty开发的网络通信程序，必须得有一个比较工业级的项目去演练一下，要不然始终就是做demo

main方法来启动，Thread.sleep()停住

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/045~055资料/055\_考虑一下网络通信中如果客户端突然异常退出应该如何处理？/笔记.docx**

《055\_考虑一下网络通信中如果客户端突然异常退出应该如何处理？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/045~055资料/052\_测试一下通过代码连接Kafka进行操作/笔记.docx**

《052\_测试一下通过代码连接Kafka进行操作》

<dependency>

<groupId>org.apache.kafka</groupId>

<artifactId>kafka\_2.12</artifactId>

<version>1.1.0</version>

</dependency>

<dependency>

<groupId>org.apache.kafka</groupId>

<artifactId>kafka-clients</artifactId>

<version>1.1.0</version>

</dependency>

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/045~055资料/051\_为IM系统部署一个示范用的单机版Kafka/笔记.docx**

《051\_为IM系统部署一个示范用的单机版Kafka》

https://archive.apache.org/dist/zookeeper/zookeeper-3.4.5/zookeeper-3.4.5.tar.gz

vi zoo.cfg

tickTime=2000

dataDir=/usr/local/zookeeper/data

dataLogDir=/usr/local/zookeeper/logs

clientPort=2181

添加环境变量

sudo zkServer.sh start

sudo zkCli.sh -server 127.0.0.1:2181

https://archive.apache.org/dist/kafka/1.0.1/kafka\_2.11-1.0.1.tgz

设置环境变量

sudo kafka-server-start.sh -daemon /Users/shishan/Documents/development/kafka/config/server.properties

kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic test

kafka-topics.sh --list --zookeeper localhost:2181

kafka-console-producer.sh --broker-list localhost:9092 --topic test

kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test --from-beginning

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/045~055资料/054\_考虑一下网络通信中如果接入系统宕机应该如何处理？/笔记.docx**

《054\_考虑一下网络通信中如果接入系统宕机应该如何处理？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/045~055资料/053\_考虑一下网络通信中如果分发系统宕机应该如何进行处理？/笔记.docx**

《053\_考虑一下网络通信中如果分发系统宕机应该如何进行处理？》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/045~055资料/049\_安装Sequel Pro对MySQL数据库进行管理/笔记.docx**

《049\_安装Sequel Pro对MySQL数据库进行管理》

在sequel pro官网直接下载即可，一般mac上都是用sequel pro进行mysql数据库管理的

CREATE TABLE `customer` (

`id` int(11) DEFAULT NULL,

`name` varchar(255) DEFAULT NULL,

`age` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/045~055资料/048\_为IM系统部署一个示范用的单机版MySQL/笔记.docx**

《048\_为IM系统部署一个示范用的单机版MySQL》

[http://dev.mysql.com/get/Downloads/MySQL-5.7/mysql-5.7.10-osx10.10-x86\_64.dmg](media/image3.png)

root账号的历史密码：SkebpENVj5\*>

vi ~/.bash\_profile

alias mysql=/usr/local/mysql/bin/mysql

alias mysqladmin=/usr/local/mysql/bin/mysql

export PATH=/usr/local/mysql/bin:$PATH

在系统偏好里有一个MySQL服务，直接启动即可

mysql -uroot -p

输入临时密码，之前生成的那个

SET PASSWORD FOR 'root'@'localhost' = PASSWORD('root');

flush privileges;

重启MySQL服务，用root账号登录，mysql -uroot -proot

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/045~055资料/050\_测试一下基于JdbcTemplate连接MySQL进行操作/笔记.docx**

《050\_测试一下基于JdbcTemplate连接MySQL进行操作》

mybatis进行一些数据库的操作

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.17</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>5.2.0.RELEASE</version>

</dependency>

springg jdbc提供的JdbcTemplate，直接可以执行SQL语句

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/045~055资料/047\_封装Jedis单例并且将认证好的session写入Redis中/笔记.docx**

《047\_封装Jedis单例并且将认证好的session写入Redis中》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/001~007资料/002\_一个IM系统的架构大致包含哪些东西？/笔记.docx**

《002\_一个IM系统的架构大致包含哪些东西？》

应用层：IM系统可以支持很多业务的，客服系统，销售系统，类似钉钉的企业内部的IM应用

API层：客服系统（电商APP，客服系统客户端模块，客服系统服务端模块）

客服系统 -> android客户端模块，嵌入在电商APP里的，在电商APP里你打开客服功能模块，相当于就是运行客服系统的android客户端模块，android客户端模块就需要跟IM系统进行连接的建立，接下来才能准备去发起一个会话，发送消息到IM系统去

客服系统 -> 服务器端模块，带着网页的，平时你看到的一些客服都是坐在电脑前面的，打开客服系统的网页，就应该要去跟IM系统建立一个连接，等待有人发送消息到IM系统，IM系统再推送给客服系统的网页上去

接入层：IM系统而言，必须得跟人家进行连接的建立，这个连接建立的过程，接收请求，返回响应，网络通信的东西，都封装在这一层

功能层：IM系统是提供很多的功能，接收消息，推送消息，群聊，红包，离线消息，安全认证，类似于这样的一些功能，很多的

存储层：两种消息存储，第一种比如说是即时性的消息，当前正在聊天的时候，大家比较关注的都是最新的一些100条消息，这些消息可以存放在Redis中；离线消息，昨天、或者几天以前、几个月以前、几年以前的消息，平时正常来说都不怎么需要看的，可以存放在MySQL数据库里，HBase大数据存储中

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/001~007资料/004\_客户端关闭、重复登录的时候应该如何处理？/笔记.docx**

《004\_客户端关闭、重复登录的时候应该如何处理？》

已经建立好连接的客户端关闭，网页关闭

一般来说玩儿这种客服系统的APP，点击联系客服的按钮，打开客服模块，跟IM系统建立连接，发送一些消息过去，从客服模块里退出来，或者是点击客服模块里有一个“咨询完毕”的按钮，直接从APP里退出，或者是长时间没有使用客服模块

代表你要跟IM系统断开连接了

主动，被动，客服模块长时间发现你没有发送消息，直接自动在底层断开连接，释放资源，直接从APP里退出，此时也会导致建立好的连接断开

用户登出的场景，跟IM系统建立好连接了，IM系统断开连接

一个用户在一个APP上已经登录了，跟IM系统建立了连接，此时他在另外一个手机上登录了一个APP，或者网页上登录了，重复登录的，IM系统要识别出来重复登录的场景，把之前登录的那个设备（APP、网页）的连接断开

重新跟最新的登录的设备建立连接，保证不要对一个用户建立多个重复的连接

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/001~007资料/005\_如何使用消息中间件让IM系统的功能模块实现解耦？/笔记.docx**

《005\_如何使用消息中间件让IM系统的功能模块实现解耦？》

基于Kafka消息中间件系统，进行了业务功能模块的解耦合

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/001~007资料/003\_IM系统的授权认证应该怎么来做？/笔记.docx**

《003\_IM系统的授权认证应该怎么来做？》

IM系统，工业级的，包含各种各样的生产级的功能模块

网络通信的，Netty来开发，TCP+自定义协议，WebSocket

安全认证，SSO单点登录，JWT安全认证的机制，在IM系统里引入授权认证

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/001~007资料/001\_IM系统都有哪些运用场景？/笔记.docx**

项目：《工业级IM即时通讯系统项目实战》

搞定底层技术，并发、网络、磁盘、IO、集合，基于这些技术直接来手撸一些中间件系统，基础架构系统，分布式海量小文件存储系统，基于Netty来项目实战，尤其是一些基础架构项目的实战

大量的运用到Netty的技术，大量的通信都是走的是长连接，必然要用Netty，TCP做长连接的开发，或者是支持Web页面（HTML5页面），WebSocket协议做长连接，Netty也可以支持WebSocket协议做长连接

缓存架构，异构存储架构，大数据量存储架构

《日上亿数据量的消息推送系统，Push系统》，第二个项目，也会大量的运用到Netty技术，架构会相对来说简单一些，做的也会快一些

分布式服务注册中心项目，给彻底搞定，全部自研，不是采用Eureka比较low的架构，网络通信环节可以用Netty来打通，做一个长连接的模式

玩儿大量的业务系统架构，中间件的源码阅读，RocketMQ、Redis、Sharding-JDBC、MyCat、Tomcat、Elasticsearch，微服务、高并发、高可用、高性能、海量数据，大规模业务系统的架构搞定他

《001\_IM系统都有哪些运用场景？》

QQ、微信、钉钉

电商类的公司和APP，IM系统，客服系统，强依赖于IM系统的，比如说你自己打开一个客户端（APP），需要跟人家的客服系统建立一个长连接，你发送消息过去，人家收到消息，人家收到消息之后，返回消息给你，一问一答

社交类的APP，探探、陌陌，跟一些人认识之后就可以进行对话，发送消息给人家，人家看到之后就可以返回消息给你，IM系统

游戏类的东西，也需要依赖于IM系统，在玩儿游戏的时候，就需要玩家之间进行即时性的交流和沟通，此时就需要IM系统，你发送一个消息给你的队友，队友返回一个消息给你

找一个在线问诊的APP，签约了一些兼职和全职的医生，我在APP上就可以付几十块钱，选择一个医生，就可以跟这个医生进行即时性的沟通，发送一些我自己的症状，照片，医生简单给我诊断一下，告诉我可以买什么药吃

IM系统，基础架构类的一个系统，是运用非常广泛的

存储层的技术，大数据领域的技术，HBase，Java类的系统里用的也是很多的

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/001~007资料/007\_IM系统中的群聊功能应该如何实现？/笔记.docx**

《007\_IM系统中的群聊功能应该如何实现？》

群聊功能是有点复杂的，很多人在一个群里，你发送了一条消息之后，别人都可以看到，别人发送了一条消息之后，你也可以看到

写扩散机制

你发送一条消息之后，是会在数据存储中写多份的，对群里的每个其他的人都会写一条数据，别人一旦登录之类的，可以查看属于自己的未读的消息，他就可以把属于自己的消息，都楼出来，自己就可以看到了

IM系统也得维护一下，创建一个群聊，一旦创建了群聊之后，就维护一条群聊的数据，里面包含了哪些人，包括群聊可以提供很多的功能，包括从群里踢人，艾特某个人，很多的，功能太多了

现在任何一个人在群里发送一条消息，写入到IM系统里，IM系统的群聊功能就得走一个写扩散的机制，把这个消息给群里所有人都写一份，群里所有人都可以在APP客户端里瞬间看到别人发送的一条消息

APP里的群聊对话框，里面很多的消息不停的弹出来，一条一条的

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/001~007资料/006\_IM系统中的单聊功能应该如何实现？/笔记.docx**

《006\_IM系统中的单聊功能应该如何实现？》

单聊功能，你一个请求发送过来，必然是会带上是要发送个谁的，是业务系统自己决定的，客服系统，销售系统，类似钉钉的IM工具

IM系统收到消息之后会进行存储，单聊功能就实现完毕了

消息如何进行推送，是由另外的一个功能模块，推送服务，从kafka里消费出来这个消息，决定是否当前要把消息推送给对方去，找到对应的连接，从这个连接中发送消息过去，就可以了

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/085\_单聊服务和群聊服务的真正面目大揭秘/笔记.docx**

《085\_单聊服务和群聊服务的真正面目大揭秘》

离线消息和历史消息，分开来做存储方案和技术方案

离线消息：如果你有百万量级的用户，每天都要频繁的登录和加载离线消息的话，离线消息的加载他是高频的行为，此时如果你基于MySQL来抗离线消息高频加载，不是太合适，MySQL不适合抗特别高的并发量

单聊服务，c2c，im-c2c-service：收单聊消息，不写任何存储，他就负责一个中转就可以了，他收到消息之后直接就把消息写回到Kafka里去

群聊服务，g2g，im-g2g-service：收群聊消息，不写任何存储，负责一个中转，他是有一些业务逻辑的，他需要查出来一个群里的所有人，对每个人都生成一条推送消息，发到kafka里去，由分发系统对群里每个人的消息推送分发给接入系统

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/096\_钉钉等国内顶尖IM系统采用的TImeline模型介绍/笔记.docx**

《096\_钉钉等国内顶尖IM系统采用的Timeline模型介绍》

Timeline模型，微信、钉钉，顶尖聊天软件，IM软件，自研分布式Timeline存储系统，来存放海量的离线消息，实现高效的多客户端消息拉取

很多个队列

对于每个receiverId都有一个属于自己的queue，timeline，基于时间轴的模型

[msg01, msg02, msg03, msg04, msg05] -> 张三的离线消息timeline队列

就不是说推送消息，推送的新消息的通知，有新的消息进入张三的timeline队列之后，此时群聊服务还是单聊服务，都是推送一个新消息通知到客户端上去。如果客户端收到了一个新消息通知之后，就会主动过来拉取消息

哪怕你当前是在线的，也会是客户端收到新消息通知之后主动来拉取

每次拉取都会带上上一次拉取消息的最大的msgId，一开始msgId=0，此时就可以从timeline队列里拉取10条数据，最大一条数据msgId=100，下次再次拉取的时候，就会从msgId=100往后去拉取10条数据

客户端会存储好自己拉取的最大的一个msgId，如果此时你下线了，有人给你发消息的时候会继续进入timeline队列里去，后面过了一天你登录了，此时你拿着昨天最后一次拉取的最大的msgId继续去timeline队列里去拉取

对于每个用户而言，只要有一个timeline队列就可以了，此时针对张三的PC端和APP端，收到的消息的msgId是不一样的，每次你在一个端登录的时候，都是拿着端自己存储的最大的msgId就去你的timeline队列里拉取就可以了

一般针对特别顶尖的IM系统，历史消息这块同样是另外一个系统，MySQL分库分表是那种中型公司是可以的，大公司，HBase、或者是自研的分布式存储系统去存放历史全量的数据也是可以的

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/100\_阿里Tablestore对Timeline模型的支持介绍/笔记.docx**

《100\_阿里Tablestore对Timeline模型的支持介绍》

IM系统，如果要用timeline模型的话，是可以采用阿里的tablestore作为存储，去存储离线消息，依托tablestore去进行消息的同步，采购了阿里云的tablestore的系统，按需付费，云的

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/089\_基于Redis SortedSet存取离线消息的技术方案细节/笔记.docx**

《089\_基于Redis SortedSet存取离线消息的技术方案细节》

写入的时候，分数就是消息发送的timestamp，按照时间自动排序

用户登录查询离线消息，采取分批查询的思想来做，发送拉取离线消息请求，必须要带上最近一次拉取过的离线消息的分数范围（也就是timestamp），因为timestamp其实是越新的数据的值越大

如果刚开始这个分数范围是空的，那么就查询SortedSet中离线消息的前10 / 20条数据，同时要带上离线消息的分数，为下次拉取来服务

拉取离线消息的时候，一般是从最旧的消息开始拉取的，大家在这里思考一下，查的时候应该怎么弄，应该是让timestamp分数从小到大排序，也就是最旧的离线消息是在最上面，然后从最旧的离线消息开始查起，查10条数据

打开一个qq、钉钉、微信，他是需要一点时间去拉取你的离线消息的

离线消息是先多一点，再多一点，再多一点，微信，有一些群，不在线的时候，群里的消息特别的多，群里的未读消息的时候一批一批加载，先加载出来群里未读消息的最旧的一批消息，立马在对话框里会再多出来一批更新一点的消息

直到把所有的未读消息都给你加载出来为止

只要客户端收到的数据不为空，就继续发起拉取离线消息的请求，再次拉取

下次拉取的时候，必须要带上上次拉取离线消息的分数范围（10条消息的起始分数和截止分数）以及拉取成功状态，此时如果发现上次拉取成功了，你就可以在这次拉取的时候把sortedset里的上次拉取的10条数据删了，就不需要了

接着再次按timestamp分数从小到大排序，再查10条数据出去

只要客户端收到的数据不为空，就继续发起拉取离线消息的请求，再次拉取，直到某次拉取，可能把上一次拉取的数据都删了，然后这次拉取发现拉到的是空的了，客户端收到的拉取响应是空，就不会再次发起拉取了

此时离线消息拉取结束

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/086\_基于Redis集群存储离线消息以及群消息写扩散的原因/笔记.docx**

《086\_基于Redis集群存储离线消息以及群消息写扩散的原因》

离线消息是属于高频的读取行为，百万用户，千万用户，亿级用户，每天会频繁的登录，下线，再次登录，拉取离线消息应该是很频繁的一个事儿，用mysql存储是不合适的，不要用mysql抗高并发

redis cluster，离线消息

redis cluster底层的原理之类的东西，亿级流量详情页系统，Jeverson要一下这个课程

无论是单聊消息还是群聊消息，离线消息服务都需要拉取

直接写入到redis cluster里去，分布式很多台机器的，是如何分散数据呢？一般来说建议按照receiverId去进行hash，让一个receiverId的离线消息都进入到Redis集群里的一台机器上去

redis里去的时候，SortedSet这个数据结构进行存储，每个recieverId会有一个SortedSet里面存放自己所有的离线消息，对于群消息，此时他应该查询出来群里有哪些人，此时他就应该走写扩散机制

对群里的每个人都生成一条离线消息，根据receiverId写入redis集群对应的机器里去

每个人receiverId都有一个sortedset，存放自己所有的离线消息，各种人发给他的单聊、群聊，未读的离线消息，都放在redis里

对于离线消息要采取写扩散呢？

一个群里有100个人，此时群里有一个人发送了群消息，此时在离线消息服务这里，对这个消息需要生成100条离线消息写入redis集群里去，分散在redis集群里的5台机器、10台机器里面的

后面的话各个人读取群消息的时候，都是从5台机器、10台机器分散去读，读取的压力不会集中在redis集群的一台机器里

但是如果说群消息仅仅是写一条在redis的一台机器里，读扩散，群里100个人，此时100个人都需要访问这1台机器去加载群里的未读消息，此时对那台机器的压力会比较大

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/084\_拉取消息的两种方式：离线消息与历史消息/笔记.docx**

《084\_拉取消息的两种方式：离线消息与历史消息》

基于netty开发的IM系统，生产级的架构核心流程都已经跑通了，demo，中大型互联网公司的自己研发的IM系统，很多都是类似的这种架构，QQ、钉钉，国内顶级的通信软件底层的IM系统去比

中型知名互联网公司，自研的IM系统，架构跟那个级别就差不多了

很多很多复杂的一些细节，那些细节我们不会带着大家动手去做了，主要是讲解思路，起码还要一个多月的时间才能动手把IM系统搞完，玩一玩netty开发生产级的网络通信程序，基于netty开发都没有问题了

netty在生产中的一些参数优化和实践，以后我们找机会来讲解的

把IM系统架构中的很多高阶的东西都给大家搞一下，讲解一下技术方案，很多技术我们之前讲过，留给大家作为作业去玩儿就可以了

IM系统，配合上我们的DFS系统，基于zookeeper完善一下他的NameNode的HA机制，即时通信 + 分布式海量小文件存储 => 跟你们公司的业务如何结合

离线消息：就是你不在线的时候，人家发送给你的未读消息，拉过去以后，QQ、微信、钉钉，登录，一些未读消息，他会用红色的字体给你一些标识，告诉你说当前可能有多少条未读消息，他会优先给你显示出来有未读消息的一些联系人的对话框

中间网络断开了，重连

历史消息：比如说你跟张三之间有一段聊天，是几天之前的，但是这几天里你跟张三又聊了很多的东西，过了几天之后，你现在要找出来5天之前的一段聊天记录，你会在你们俩的对话框里，不断的向上滚动聊天框，他就会不断的加载出来你们俩之前的聊天记录，一直往上滚动，一直就可以加载到5天之前的聊天记录

消息漫游，假设QQ会在客户端本地缓存几天的消息，供你查阅历史消息，此时就只能从云端，服务端，去拉取历史消息

比如说你换了一台手机，或者在一个新的电脑上登录了你的QQ，此时默认情况下，你对一个对话框往上翻，其实是翻不到历史消息的，此时你需要打开聊天记录的面板，在里面可以做一个消息漫游，把2年内的消息都拉取到客户端本地来

此时你就可以在客户端本地查阅2年内的历史消息，甚至可以对历史消息进行搜索，拉取到客户端本地的时候回用lucene这样的全文检索框架，建立倒排索引，供你来搜索

充值vip，类似于这样的一些场景，查阅已经读过的历史上的消息，历史消息

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/101\_中小型公司到底要不要采用复杂的Timeline模型？/笔记.docx**

《101\_中小型公司到底要不要采用复杂的Timeline模型？》

中小型公司，如果真的是自研IM系统的话，不用timeline模型也是可以的，客服系统，可能并不存在说多个端的登录，销售系统，用户量就几万人，或者是几十万人，属于大数据领域范围的一个系统，hdfs

HDFS，HBase，才能去玩儿他，非常的复杂，性能，故障，很多东西你可能hold不住

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/092\_整体梳理一下离线消息和历史消息的存取技术方案流程/笔记.docx**

《092\_整体梳理一下离线消息和历史消息的存取技术方案流程》

还应该查一下群里每个人的入群时间，必须是这条群消息的发送时间是在入群时间之后，对于这种群里的人，此时会对群里每个人都生成一个消息推送

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/095\_一个技术细节补充：利用snowflake算法生成messageId/笔记.docx**

《095\_一个技术细节补充：利用snowflake算法生成messageId》

messageId是如何生成的呢？直接插入数据到mysql里，messageId是自增长的主键，但是现在我们的messageId是不能这样做了，一条消息在请求分发系统的时候，就自动生成一个唯一的messageId

比如基于redis自增长，但是怕就是怕万一redis挂了；另外就是让请求分发系统去不停的插入一个自己的id表，自增长id生成；snowflake算法，面试系列里讲过，就是用他来生成唯一的id

就是可以作为消息的messageId

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/090\_用户在线时消息推送成功过了，离线消息如何处理？/笔记.docx**

《090\_用户在线时消息推送成功了，离线消息如何处理？》

单聊 or 群聊，如果目标用户在线，就会自然推送成功，此时可能离线消息还是存储了消息在Redis中，那么直接这样的话，会导致用户已经读到的消息，下次登录会显示为未读离线消息，这是不行的

所以说单聊 or 群聊，只要用户在线推送成功了，此时收到最终响应，不是跟以前一样，把mysql的is\_delivered设置为1，而是应该根据消息的timestamp作为分数，直接去redis的sortedset中删除离线消息

这样下次用户登录就不会加载已读的消息为离线消息了

redis cluster务必进行容量预估，每天那么多活跃用户下，在redis中存放的离线数据会由多少，必须是你读取过了一些消息之后，然后才能保证说可以从redis cluster里删除你的离线消息

离线消息量过大，导致redis cluster中的内存满了，此时就会很尴尬，可能导致redis自己用LRU算法自动清理掉一些离线消息数据，你在登录的时候，有一些离线消息就加载不出来了

尽可能部署一个大规模的redis cluster去盛放你所有的离线消息数据

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/093\_多客户端同步消息的时候为什么要写多份离线消息？/笔记.docx**

《093\_多客户端同步消息的时候为什么要写多份离线消息？》

多客户端

比如我是石杉，我是一个人，我平时用的比较多的客户端就是APP

我经常跟一个叫做张三的聊，张三，他平时手机上的APP会打开，电脑上的PC端聊天软件也会打开，张三他一个人会同时登陆多个端，手机端，PC端

我给张三发送消息

张三的APP端和PC端可能是连接到不同的接入系统的

离线消息服务需要为每个用户的每个端都存储一个SortedSet，你在不同的端上收到的消息可能是不一样的，APP端你今天登录了，PC端今天没登录，APP端就会收到一些离线消息，PC端会积累更多的离线消息

明天当你登录PC端的时候，你需要收取很多的离线消息

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/099\_Timeline模型在落地实现中的一些技术难点/笔记.docx**

《099\_Timeline模型在落地实现中的一些技术难点》

Redis之类的，并没有一个非常成熟的支撑Timeline模型的分布式存储系统，又要可以支撑海量数据分布式存储，又要支持timeline模型，高并发、高性能，离线消息数据没有必要长期存储的

比如张三有一个timeline队列

[msg01, msg02, msg03, msg04, msg05] -> 两个端，APP端，PC端

APP端已经读取到了msg04，PC端登录读取到了msg05，在msg04之前的三条数据，msg01~msg03，那三条数据就可以删除掉了

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/087\_Redis的SortedSet数据结构是怎么用来存储离线消息的？/笔记.docx**

《087\_Redis的SortedSet数据结构是怎么用来存储离线消息的？》

im-c2c

im-g2g

im-offline-message：离线消息服务

receiverId都有一个sortedSet

比如说按照每条消息发送出来的timestamp，时间戳作为分数

zadd offline\_messages\_#{receiverId} #{score} #{message}，按照顺序，分数从小到大

zrange offline\_messages\_#{receiverId} 0 -1，从小到大

zrevrange offline\_messages\_#{receiverId} 0 -1，从大到小

0~5，从小到大取出来前5条数据，从大到小取出来前5条数据

withscores，同时取出来分数

zrangescore offline\_messages\_#{receiverId} -inf #{某个时间戳}

可以去获取截止到某个时间戳内的离线消息

比如说，一个用户现在登录了，此时：zrange offline\_messages\_#{receiverId} 0 -1 withscores，获取出来当前所有的离线消息，此时他是知道最近一条消息的score

下次获取消息的时候，zrevrangescore offline\_messages\_#{receiverId} -inf #{上一次获取的数据最大的score}，此时一直获取截止到上一次获取的数据里最大的那个socre为止，小于那个score的数据，我们就不要了

没有直接动手实验，也许我讲的命令不一定完全准确，大家可以自己去搞一些实验类的数据，测试一下，这个都是无所谓的

你可以定期的从sortedset拉取数据的时候，定期的就把已经拉取过的离线消息都从sortedset里删除掉，保证sortedset里的数据不要太多，避免把redis的内存搞的爆满了，对不对

zremrangebyscore offline\_messages\_#{receiverId} #{起始分数} #{结束分数}，此时可以删除掉一些已经读取过的离线消息

zrank，告诉你某条数据在sorted set里的序号

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/094\_多客户端同步时写多份离线消息的存储成本问题/笔记.docx**

《094\_多客户端同步时写多份离线消息的存储成本问题》

如果我们要支持一个用户登录多个端去接收消息的话，我们就需要对一个用户的多个端写多份离线消息，redis cluster，内存，缓存来存放离线数据的，内存本来就是有限的，远远比磁盘空间要少的多

为了支持多个端，此时还要搞多份离线消息，相当于是为了支持多个端，瞬间让redis cluster里的数据量暴增了1倍，主要是用他的一些特殊的数据结构实现高并发的读写，分布式集群的存储

离线消息的量比较大的话，可以不用redis cluster来存，mongodb，基于磁盘的分布式NoSQL数据库来存，性能比较高，并发能力也比较强，还基于磁盘来存放数据，你就不用考虑过多的数据存储的消耗了

如果说确实要基于redis cluster来存，redis集群多部署一倍的机器

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/098\_针对Timeline模型采用的Hash分发机制保证消息顺序性/笔记.docx**

《098\_针对Timeline模型采用的Hash分发机制保证消息顺序性》

我们对每个请求都有一个头字段，一直没用上，sequenceId

在客户端层面，他在发送消息的时候，其实一般必须是自己自增长去生成sequenceId，对每个请求的sequenceId都是自增的，代表了消息的顺序

hash分发，把相同的receiverId的消息都分发给一个分发系统的实例

对每一个receiverId进行一个hash，对分发系统的数量进行取模，就可以保证说同一个receiver的消息都是分发给同一个分发系统，同一个partition内部是有序的，分发系统投递消息到kafka的时候，要按照receiverId去hash路由到同一个partition里去

就是同一个receiver的消息有序的进入他的timeline模型里面去，每条消息都有自己的sequenceId，加载到消息的时候，此时就可以根据sequenceId来判断一下，消息展示出来的sequence必须是增长的

如果出现有sequence乱序，此时你客户端那边重新调整一下顺序去展示就可以了

客户端在发送消息的时候，sequenceId不一定是数字，字符串

石杉，张三和李四发送消息

发送给张三的sequenceId：132\_153\_13546，132\_153\_13547，132\_153\_13548

发送给李四的sequenceId：132\_266\_8990，132\_266\_8991

客户端的层面首先是保证了一个顺序的，是有顺序的

假设说在传输数据的过程中不知道为什么，有的消息丢失了，此时一定不用担心，消息有两份，timeline系统那儿有一份，在历史消息系统那儿有一份最完整的数据，那里的数据一定是不会丢的

张三收到的石杉发送的消息的时候，可以检查他的sequenceId

132\_153\_13546

132\_153\_13548

可以做一个客户端的补偿机制，自动发送请求去历史消息服务里查询，132\_153\_13647，这个sequenceId的数据

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/088\_历史消息的分库分表技术方案以及群消息读扩散的原因/笔记.docx**

《088\_历史消息的分库分表技术方案以及群消息读扩散的原因》

离线消息服务 + 写扩散 + Redis Cluster + receiverId hash + SortedSet

历史消息服务

低频，一般大部分的用户主要是读每天的未读消息，在线，就会即使的读到人家推送过来的最新单聊或者群聊，如果登录，未读和离线消息。一般对某个对话框往上翻，查阅历史消息是属于低频的行为

我们就没有必要使用Redis了，离线消息，只要是被你接收了，此时就可以从Redis里删除了，所以离线消息数据量比较少，读取高频，高并发、高性能，所以非常适合采用Redis集群来实现

历史消息，他低频，查阅数据的量可能很大的，他要从全量数据里去查找消息

MySQL分库分表的方式来实现，面试突击第一季里，Sharding-Sphere（Sharding-JDBC），MyCAT，可以自己看一下，分库分表中间件

单聊消息：senderId，按照发送人id，来进行分库分表，把一个单聊消息表，按照senderId水平拆分，分到多个数据库上去，多个表中，用多台数据库来存放海量的消息数据

同一个senderId发送的单聊消息一定是在一台数据库的一个表中的

群聊消息：读扩散，不是写扩散，一条群聊消息就写一条数据，不会生成多条数据，按照groupId，群id进行分库分表，水平拆分，所有的群聊数据拆散在多台数据库服务上，多个表中，用多台数据库服务器来存储海量的群聊消息

同一个groupId发送的群聊消息一定是在一台数据库的一个表中的

你找一些对话框去往上翻聊天记录，查阅历史消息，或者进行消息漫游，都是针对某个对话框，一个人或者是一个群，都是针对的是一个对话框去进行的历史消息的查询

假设你现在要翻越你跟张三之间的历史消息，此时这个对话框里需要查找的是你发送给张三的历史消息，以及张三发送给你的历史消息，比如说根据senderId=你自己，去路由到一个数据库上去，再根据senderId=你自己路由到一个表里去，在这个表里查senderId=你自己，receiverId=张三的所有数据，按照时间范围来查

你就可以查找出来所有历史上你发送给张三的数据

再根据senderId=张三，路由到一个库上去，再路由到一个表里去，查找senderId=张三，recieverId=你自己，按照时间范围来查，张三发送给你的所有的数据，两份数据合并一下，此时就是你跟张三的对话框里的所有的历史聊天记录

如果你翻开一个群，要查询群的历史消息，此时就根据groupId路由到一个库里去，再路由到一个表里去，在这个表里可以找到groupId所有的数据，此时按照时间范围查询群里你要查阅的历史消息就可以了

存放了这个群里各种不同的人发送的所有的历史消息

你随便找一个分库分表中间件 ，sharding-jdbc，mycat，建立多个库，里面分很多表，按照我说的这套方案，根据不同的senderId或者是groupId，分库分表写入数据进去，查询的时候就是根据我说的方案路由和查询

读扩散，不是写扩散，写一条，根据groupId分库分表，群消息表，g2g\_message，c2c\_message

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/103\_接下来IM系统的架构设计 还有哪些东西要讲解？/笔记.docx**

《103\_接下来IM系统的架构设计还有哪些东西要讲解？》

认证，JWT去优化

HTTP加密通信

IM系统的消息加密存储

发红包

会话列表

图片和文件的优化

多级缓存架构

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/097\_Timeline模型下可能存在的消息顺序错乱以及丢失问题/笔记.docx**

《097\_Timeline模型下可能存在的消息顺序错乱以及丢失问题》

msg08，msg09

[msg01, msg02, msg03, msg04, msg05, msg06, msg07, msg09, msg08, msg10, msg11]

就会导致第一消息顺序乱了，第二加载的时候

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/102\_基于Kafka构建松耦合IM系统架构的优点/笔记.docx**

《102\_基于Kafka构建松耦合IM系统架构的优点》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/084~103资料/091\_基于MySQL分库分表存取历史消息的技术方案细节/笔记.docx**

《091\_基于MySQL分库分表存取历史消息的技术方案细节》

存的时候就是单聊和群聊两个维度进行分库分表，读历史消息的时候思路也讲过了，但是有一个点，就是用户在群里不是可以拉取无限制的历史消息的，因为我们得有一个拉群信息表，记录用户加入群的时间

然后在拉取群消息的时候，必须确保这个用户不能拉取早于他入群时间的消息，这个是一个很关键的细节

拉取历史消息的时候同样是分批加载

可以限定比如一次可以加载100条数据，一次历史消息加载请求就是最多查100条消息，查历史消息应该是按照时间从大到小排序后加载，比如先查距离现在最近的100条消息，然后下次你继续查询，查询的时候带上上次加载的消息最小时间

接着加载小于上次最小时间的更旧的100条消息

直到最后加载不出来了，就停止历史消息的加载，基本是这个意思

QQ消息漫游也是一样的，比如说我在mac上一台新电脑，登录之后，跟很多人很多群之前的聊天记录都是加载不出来的，本地没有缓存。有的时候要一下子漫游同步历史2年的数据到客户端，此时就是分批加载分批加载，直到全部加载完成

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/068\_业务系统在存储完单聊消息过后将消息推送写入Kafka中/笔记.docx**

《068\_业务系统在存储完单聊消息过后将消息推送写入Kafka中》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/079\_进行单聊消息功能的全流程测试/笔记.docx**

《079\_进行单聊消息功能的全流程测试》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/071\_客户端SDK接收到消息推送/笔记.docx**

《071\_客户端SDK接收到消息推送》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/083\_一个课后作业：自己实现离线消息拉取功能/笔记.docx**

《083\_一个课后作业：自己实现离线消息拉取功能》

比如有一个人发送消息给别人，或者是一个群，此时接收消息的人不在线，也就是说跟接入系统并没有建立连接，此时就不会反向推送消息过去了

但是只要你一旦登录了之后，此时就会自动发起一个拉取离线消息的请求，就会尤其是把message\_receive表里属于你的未读的消息，is\_delivered=0的消息，推送给你，你就可以看到一些未读的消息

另外，如果在聊天框里不停的上拉聊天框，就可以不停的发送请求加载他的聊天记录，这个 东西完全是可以做成对外提供的是http接口，web系统，人家发送请求，你就直接调用业务逻辑系统的rpc接口（spring cloud，dubbo），加载一批消息出来

单聊，群聊，离线消息拉取，这三个东西其实是IM最最核心的功能，IM系统，即时通信系统，核心功能支持这三个就可以了

一个人发送一个文件给另外一个人，他也是基于IM来实现的，聊天客户端，文件先要保存到分布式海量小文件系统里去，然后把文件的id，消息类型，通过IM系统去保存起来，然后IM系统会把文件id，文件名，消息类型（file），反向推送给另外一个客户端

另外一个客户端上的聊天软件业务代码，渲染在对话框里，显示的是一个文件名

如果你点击这个文件，就是根据文件id，从分布式海量小文件系统里去下载一个文件出来就可以了

聊天软件，加好友、建群拉人，单聊，群聊，发文件，云盘

聊天软件业务系统，IM系统，DFS系统

QQ空间，礼物，红包，视频，语音，之类的一些东西，聊天软件

传统公司，IM系统支撑了销售系统，CRM系统；DFS系统支撑了业务文件的存储

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/078\_为最终的测试准备好MySQL、Redis和Kafka/笔记.docx**

《078\_为最终的测试准备好MySQL、Redis和Kafka》

kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic send\_message

kafka-topics.sh --list --zookeeper localhost:2181

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/069\_分发系统收到消息推送后转发给接入系统/笔记.docx**

《069\_分发系统收到消息推送后转发给接入系统》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/076\_重新梳理消息发送的全流程以及加入调试用的日志输出/笔记.docx**

《076\_重新梳理消息发送的全流程以及加入调试用的日志输出》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/075\_业务逻辑系统收到消息推送响应之后更新消息的投递状态/笔记.docx**

《075\_业务逻辑系统收到消息推送响应之后更新消息的投递状态》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/073\_接入系统将消息推送的响应转发给分发系统/笔记.docx**

《073\_接入系统将消息推送的响应转发给分发系统》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/080\_调试单聊消息全流程测试中发现的问题/笔记.docx**

《080\_调试单聊消息全流程测试中发现的问题》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/081\_继续测试单聊消息功能的全流程/笔记.docx**

《081\_继续测试单聊消息功能的全流程》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/072\_客户端对消息推送返回响应给接入系统/笔记.docx**

《072\_客户端对消息推送返回响应给接入系统》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/074\_分发系统将消息推送的响应转发到Kafka中去/笔记.docx**

《074\_分发系统将消息推送的响应转发到Kafka中去》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/082\_一个课后作业：自己去实现类似的群聊功能/笔记.docx**

《082\_一个课后作业：自己去实现类似的群聊功能》

基于Netty技术如何实现长连接，其实就是可以做到用户和系统之间的双向通信，系统可以反过来去推送数据给用户，也可以做到系统和系统之间的双向通信，Netty技术是非常非常的常用的

crud，技术含量真的很低，netty这个技术是大家成长为技术高手必备的东西

群聊，跟单聊是类似的

必须有一个表，里面有群的记录，可以允许用户建群，在群里拉人，踢人，之类的，实现一个这样的功能，不需要使用TCP长连接的方式来做，做一个业务系统，提供HTTP的接口也是可以的

spring web mvc，做一个web系统

比如有用户在群里发送一个消息，消息一层一层过来，到业务逻辑系统那里，除了在message\_send里写入一条，要在message\_receive里写入多条数据，对群里的每个人都要写入一条数据

业务逻辑系统要对群里每个人都发送一个消息push到kafka去，由分发系统把不同的人的push消息分发给不同的接入系统，根据redis里的session就可以判断了，接入系统把消息反向推送给对应的人的客户端的sdk

聊天业务客户端的代码，此时就可以把这个消息渲染到群聊天记录里去

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/077\_重新梳理消息推送的全流程以及加入调试用的日志输出/笔记.docx**

《077\_重新梳理消息推送的全流程以及加入调试用的日志输出》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/068~083资料/070\_接入系统将消息推送反向转发到客户端SDK上去/笔记.docx**

《070\_接入系统将消息推送反向转发到客户端SDK上去》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/067\_SDK收到发送单聊消息的响应/笔记.docx**

《067\_SDK收到发送单聊消息的响应》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/060\_TCP接入系统收到单聊消息之后转发给分发系统/笔记.docx**

《060\_TCP接入系统收到单聊消息之后转发给分发系统》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/057\_为单聊功能设计数据库表结构/笔记.docx**

《057\_为单聊功能设计数据库表结构》

很多同学一开始问我，IM系统的表结构？屁股想想都知道了，很多事情不要过多的依赖于我，我说的一定是权威的

核心表就是两个表就可以了

消息发送表：每个人发送了哪些消息都记录在里面

消息接收表：就是你的这个消息发送出去是给谁，就记录在这个表里

群信息表，有哪些群

群关系表，每个用户跟群的关联关系

mysql跑起来，启动数据库管理软件，seqel pro，建好两个表

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/061\_分发系统将单聊消息写入Kafka中/笔记.docx**

《061\_分发系统将单聊消息写入Kafka中》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/056\_设计一下单聊功能的业务流程/笔记.docx**

《056\_设计一下单聊功能的业务流程》

Java架构和大数据架构，不分家，有时间有精力，两个方向一块儿学

用一些IM系统，发送消息的时候，发出去一个消息，如果网络不太好，消息旁边会显示一个“发送中”的标志，变换一个颜色，意思就是已经发送出去了

单聊做好了，群聊很多代码是类似的

你在一个群里发送一个消息，然后这个消息过来，会自动在数据库写入群里各个成员的消息接收表

然后把这个消息反向推送个群里各个成员就可以了

业务模块，客服系统，聊天系统，负责把消息渲染在群聊聊天框里就可以了

离线消息，每次你一登录，就自动把你的一些消息给显示出来就可以了， 尤其是你还没有读过的一些消息，直接从mysql里查询就行了

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/066\_TCP接入系统将单聊消息响应转发回客户端/笔记.docx**

《066\_TCP接入系统将单聊消息响应转发回客户端》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/063\_完成业务系统将单聊消息写入MySQL的代码/笔记.docx**

《063\_完成业务系统将单聊消息写入MySQL的代码》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/059\_在SDK中设计一个发送单聊消息的接口/笔记.docx**

《059\_在SDK中设计一个发送单聊消息的接口》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/058\_为单聊功能设计Protobuf协议消息体/笔记.docx**

《058\_为单聊功能设计Protobuf协议消息体》

protoc –java\_out=im/src im/proto/AuthenticateRequest.proto

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/064\_业务系统将单聊消息存储响应写回到Kafka/笔记.docx**

《064\_业务系统将单聊消息存储响应写回到Kafka》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/065\_分发系统从Kafka获取单聊消息响应转发给接入系统/笔记.docx**

《065\_分发系统从Kafka获取单聊消息响应转发给接入系统》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/056~067资料/062\_业务系统将单聊消息写入MySQL中进行存储/笔记.docx**

《062\_业务系统将单聊消息写入MySQL中进行存储》

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/143\_后续课程的预告：其实以后我会带大家一步一步撸出来/笔记.docx**

《143\_后续课程的预告：其实以后我会带大家一步一步撸出来》

IM系统，也会明年抽时间，我会把现在讲的所有的架构，都一步一步写出来，录制成视频，到时候大家也可以去看一下，参考一下，自己先做一遍

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/129\_针对服务端推送消息丢失问题的ACK解决方案/笔记.docx**

《129\_针对服务端推送消息丢失问题的ACK解决方案》

推送一条消息之后，维护一个等待ACK列表，如果一段时间后没有收到ACK，那么就需要进行重试，保证客户端可以收到，客户端需要自己进行去重，避免重复显示消息

单聊服务 -> 消息投递系统

对于客户端，他需要在自己的本地进行消息的去重，判断一下某个sequenceId的消息是否已经被显示出来了，不要重复的区显示同一个sequenceId的消息

ACK + 重试 + 去重

单聊服务不一定知道消息是否在线的，如果说客户端不在线，肯定就是没收到消息是正常的，能感知到客户端是否在线的

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/127\_再看IM系统消息传输链路中的数据丢失问题/笔记.docx**

《127\_再看IM系统消息传输链路中的数据丢失问题》

发送消息、消息存储以及消息推送，三个环节

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/133\_使用HttpDNS解决LocalDNS被劫持的问题/笔记.docx**

《133\_使用HttpDNS解决LocalDNS被劫持的问题》

客户端 -> LocalDNS -> DNS，不走

客户端 -> HttpDNS服务 -> DNS，HttpDNS服务基于Http协议，而不是DNS的协议，不会被劫持，现在一般云厂商都提供HttpDNS服务

安全架构，加密通信、黑客攻击、网络攻击、数据泄密、账号泄漏、加密存储、加密算法，一整套的加密方案，安全性

HttpDNS的简单的原理，以及如何基于云厂商购买这个服务和部署，配合IM系统来域名解析的使用，留个作业给大家，自己可以深入的了解一下，方案了解清楚了，出去面试，起码在安全性这块，你也可以说一说

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/138\_发送语音时是如何让对方无需下载立刻可以听到的？/笔记.docx**

《138\_发送语音时是如何让对方无需下载立刻可以听到的？》

语音一般来说都有时间限制，比如说1分钟，一般来说是一个很小的文件，几MB的语音文件，如果走分布式文件系统来传输和存储语音文件，人家看到一个语音消息之后，点开这个消息，此时需要几秒钟的时间去下载这个语音文件

体验很不好

把语音拆分成多个数据分片，传输到服务端，然后反向推送过去，在接收方合成之后再让他收听

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/128\_针对客户端推送消息的丢失情况的ACK解决方案/笔记.docx**

《128\_针对客户端推送消息的丢失情况的ACK解决方案》

客户端推送消息丢失了，在一段时间内没有收到服务端存储消息后返回的应答，超时认为异常，此时重试，但是可能有重复，此时需要对每条消息都有一个唯一的requestId，给服务端进行去重

针对消息的发送，到底处理到什么程度，可以认为一次消息发送成功了呢？

必须是这条消息被落地到存储中之后，才可以认为这个消息投递成功了，历史消息服务落地数据到mysql中持久化存储之后，此时就可以认为这个消息一般来说不会丢了，历史消息服务回传一个ACK的response的消息

等待ack的列表里去，客户端，超时时间，如果超过5s还没收到ack的response，此时就进行重发，钉钉、QQ、微信，如果他们因为一些网络原因或者是问题，发现消息传输不成功，此时会在你的那条消息旁边，给你一个红色的小按钮

此时你可以选择手动点击那条消息的红色小按钮，他会重新尝试去发送

如果你消息没有发送成功，此时你的客户端比如被你卸载了，删除了本地所有的缓存数据，此时没发送成功的消息就会丢失

如果说有重发，就有可能导致消息的重复

sequenceId，实际上来说是客户端生成的一个唯一标识一个消息的id，设计在表结构里的，在历史消息服务进行存储的时候，包括单聊服务，都可以基于这个sequenceId进行去重，各个服务，离线消息服务，历史消息服务，单聊服务，你在收到一个消息之后，决定对他处理之前，都应该根据sequenceId判断一下

之前是否已经处理过，此时就可以进行去重，不要再次处理了，都可以根据这个id结合redis之类的东西来进行去重，实际上来说都会在mysql里会进行更新，去更新他的推送成功的字段这样子

ACK + 重试 + 去重

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/131\_如果服务端宕机导致没有推送消息该如何解决？/笔记.docx**

《131\_如果服务端宕机导致没有推送消息该如何解决？》

如果服务器宕机了，导致消息没能推送给客户端，怎么办？此时是没有关系的，因为没推送成功，就不会收到ACK，离线消息一直存在，后续客户端登录会自动加载离线消息的，所以是没有问题的

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/135\_IM系统中的消息数据加密问题/笔记.docx**

《135\_IM系统中的消息数据加密问题》

在服务端采用一套复杂的加密机制来进行加密，但是这里我们不展开了，因为涉及到Java加密相关的解决方案，非常复杂

也可以是端到端加密，就是两个客户端生成密钥对，交换公钥

发送客户端用接收客户端的公钥加密，服务器端接收到的是加密数据，因为没有人家的私钥，无法解密

只有加密数据发送到接收客户端，人家用私钥才可以解密

这个对性能损耗是很大的，国内用的不多，一般还是服务端加密

留一个作业，自己上网找一些第三方的专业的加密厂商，了解他们的一些方案，根据那些方案，可以考虑如何用到自己的IM系统里去

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/125\_未读数功能可能存在的数据不一致问题/笔记.docx**

《125\_未读数功能可能存在的数据不一致问题》

未读数，可能会存在一定的不一致的问题

在redis里存储一个用户所有的未读数，很多个key，总未读数是一个key，跟每个人的会话的未读数都是一个key，当你有新的未读消息的时候，会涉及到多个key的更新，有多个key需要同时进行更新

如果说总未读数+1成功了，但是跟某个会话的未读数+1失败了，此时就会出现数据不一致的问题，一旦说数据不一致的话，可能就会导致如果你在某个新的客户端登录之后，此时你加载到的未读数发现是不一致的

比如总未读数是10，结果各个会话的未读数加起来只有8，此时你会觉得很疑惑

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/140\_如何通过CDN以及预热机制来加速大视频与图片的下载？/笔记.docx**

《140\_如何通过CDN以及预热机制来加速大视频与图片的下载？》

百度一下他的概念

这个客户端的人，他人在北京，我们的DFS系统的服务器部署在上海的机房，此时北京到上海的距离是很远的，我们可以去购买一些厂商提供的北京的CDN服务

CDN的基本原理，看看阿里云，腾讯云，提供的CDN的服务，以及如何使用

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/136\_图片以及视频是如何基于分布式文件系统来存储的？/笔记.docx**

《136\_图片以及视频是如何基于分布式文件系统来存储的？》

企业级的IM系统给他都做出来，还是挺耗费时间的

消息的内容，就是文件的名字，或者文件的id

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/126\_引入分布式锁技术保证未读数的准确性/笔记.docx**

《126\_引入分布式锁技术保证未读数的准确性》

对于这个未读数功能他的一个更新，必须要引入一个技术，分布式锁

出去面试的，系统里为什么要用分布式锁？MySQL提供的行锁？因为在实际的系统开发中，又不是只有数据库的更新，分布式缓存的更新、分布式文件系统的更新、ES/MongodB等NoSQL的更新

在一个业务逻辑中，先在MySQL中更新一些数据，然后在Redis中更新一些数据，接着在ES里更新一些数据，此时如果你要保证你对一个数据进行更新的时候，其他人不能同时更新，避免数据错乱，此时只能用redis/zk提供的分布式锁的功能

未读数的更新，总未读数和某个会话的未读数的更新，在这个过程中，必须得加锁，如果不加锁的话，有可能会在分布式的场景下，出现一些数据错乱的问题。有可能会出现，在同一个时间，未读数服务的多台不同的机器，都想对同一个用户的未读数进行更新，加，扣减，如果你不加控制，可能会导致数据的错乱

比如现在有未读数服务的两台机器，同时在更新李四的未读数

总未读数：0

和张三的会话的未读数：0

第一台机器：加锁，总未读数+1（成功）-> 1，张三会话的未读数+1（失败）-> 0，此时需要对张三会话的未读数进行重试，必须让他执行成功之后，别人才能继续对李四的未读数进行更新，张三会话的未读数+1（重试成功）-> 1，释放锁

第二台机器：被锁阻塞住，获取到锁，总未读数-1（成功）-> 0，张三会话的未读数-1（成功）-> 0

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/141\_如何通过缩略图以及高清大图的机制来提升图片加载性能？/笔记.docx**

《141\_如何通过缩略图以及高清大图的机制来提升图片加载性能？》

传输一个图片

对于视频播放的优化

如果说打开一个人家传输的视频，打开了以后才开始下载这个视频，此时一定会等待很长的时间，这个效果也是很差的

视频的前10秒，前20秒的小视频，截取，5分钟，30秒的视频截取出来，点开立马可以秒播，10s，立马可以播放了，接下来同时客户端采用流的方式从dfs系统不断的去读取数据流过来

读取到一些视频数据流，就可以播放一些出来

语音分片上传，传输完毕再显示；图片的缩略图机制；视频的前n秒缓冲机制，就可以提升他们的展示性能

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/124\_IM系统中的未读数功能应该如何来实现？/笔记.docx**

《124\_IM系统中的未读数功能应该如何来实现？》

微信、QQ、钉钉，可以给你显示有多少消息未读，跟某个人的会话有多少条消息未读

有消息过来就增加未读，全局和会话两个维度，有收消息的应答，就扣减未读消息数量，对外提供HTTP接口供其查询即可，主要针对多端场景

其实每个端自己本地也得存一份，有消息来就增加，然后用户看了就扣减，如此即可

张三 -> 李四，发送了一条消息

他就需要给李四的总未读数+1，李四和张三的会话的未读数+1，这块东西，不用解释，直接用redis来实现就可以了

如果发现kafka里有，李四确认收到张三消息的通知之后，此时可以给李四的未读数-1，李四和张三的会话的未读数-1

在服务端去存储一下，客户端自己得维护未读数的统计，收到一条张三的消息之后，在客户端本地给未读数+1，和张三的会话的未读数+1；当客户端发现李四阅读了张三发送过来的消息之后，就给总未读数-1，张三的会话的未读数-1

为什么客户端和服务端都要维护一份未读数呢？

在iphone的手机上登录了，此时iphone本地的APP，维护一份未读数，在服务端也维护了一份未读数

此时你切换到华为的手机上登录APP，华为本地是没有未读数的记录的，此时他必须在一个新的客户端登录之后，发送http请求到服务端加载当前的未读数，另外加载一部分的历史记录出来，加载一些离线消息出来

登录新的端，可以从服务器那边加载未读消息数

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/134\_再看数据传输过程中面临的几点安全隐患/笔记.docx**

《134\_再看数据传输过程中面临的几点安全隐患》

连接断开、截取数据、篡改数据、伪造数据

如果网络被攻击，导致某个接入机器的连接都断开了，此时可以从iplist服务选择其他的接入机器就可以了

对于黑客在数据传输过程中的截取、篡改和伪造，我们是基于TCP自定义的私有协议，只有我们自己知道这里面的内容的含义，大家可以自己思考一下，别人是不知道的，不像HTTP那种，是很容易截取和篡改的，大家都知道HTTP协议传输哪些东西

4个字节是什么，4个字节是什么，8个字节是什么，一大堆的字节拼接出来一套数据，自定义协议，只有我们知道如何根据这个协议反向解析出来对应的数据

通用协议，http协议，xml协议，webservice，http

header，每个header是什么意思，哪些数据是body，他其实都知道

另外我们还可以通过SSL加密，对数据进行加密，就没法被人获取到数据

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/130\_用来唯一标志一条消息的SequenceID到底应该如何生成？/笔记.docx**

《130\_用来唯一标志一条消息的SequenceID到底应该如何生成？》

不丢，ACK + 重试，去重 -> sequenceId（客户端生成的）

其实对于一个客户端每次跟另外一个客户端之间的对话都是一个会话，然后在这个session内发送的消息都是有一个递增的SequenceId

站在张三的角度，跟李四的是有一个会话，对于这个会话你是不是可以生成一个sessionId，唯一标识张三角度跟李四的一个会话，在这个会话里张三自己每次发送出去的消息，都是一个sequenceId，递增的

通过这个sessionId + sequenceId，其实就可以唯一标志一条消息，在服务端进行去重，同时在客户端进行去重了

当然也可以由客户端自己保证，不同的session的sequneceId不会重复，全局递增，但是这个有点难以保证，毕竟客户端可能会清空一些本地存储的数据，最好还是跟sessionId绑定起来去保证

如果不想使用两个id，可以用那种64位的long型id，把高32位作为sessionId，低32位作为sequenceId，也是可以的，这样就用一个sequenceId包含了高低32位即可，其实大家到这里就理解我们之前一直传递的sequenceId是用来干什么的了

字符串来处理，sequenceId字符串，“sessionId\_sequenceId”

sessionId和sequenceId，都是存储在客户端，如果客户端被卸载和删除了本地缓存数据之后，那怎么办呢？对于sessionId和sequenceId，可以存储在服务端一份，生成一个sessionId，可以发送http请求到服务端来请求生成一个唯一的

每次服务端收到一个客户端发送的消息的sequenceId，代表了当前最新最大的一个sequenceId，服务端一定会存储好这个客户端跟另外一个客户端的会话中的最新最大的一个sequenceId

sequenceId递增的，对方客户端收到的消息都可以自动根据sequenceId去进行排序，保证消息顺序不会错乱

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/139\_拆分大视频与语音过后的断点续传问题/笔记.docx**

《139\_拆分大视频与语音过后的断点续传问题》

只是一个小视频，其实直接加载视频文件到内存里，一次性上传给dfs系统，上传失败了，重试，对于大视频，采取流的方式去传输，一点点给人家上传

对于客户端必须记录下来当前对一个大视频或者语音，当前上传到了哪个地方，对于你上传到的这个地方，万一上传的过程中网络断开了，下次网络重新连接起来的时候，你要进行一个端点续传

对于服务单而言也得支持端点续传

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/137\_分布式海量小文件存储系统可以放大视频吗？/笔记.docx**

《137\_分布式海量小文件存储系统可以放大视频吗？》

如果只是一些图片的话，放我们之前做的分布式海量小文件存储系统是没有什么问题的，如果说是一些大视频呢？比如几十MB，几百MB，几个GB的超大文件，或者超大视频呢？答案应该是可以的

使用流的方式把超大的文件传输给我们的后台系统这样子

直接把文件加载到内存里去，传输给后台系统，一边读取大视频的字节流，一边把字节流传输到文件系统上去，几个GB的超大文件在里面

hdfs，大文件分片存储，几十MB的小视频，几百MB的视频，一两个GB一般都是一个大的电影，微信拍摄视频发送给别人，一般来说微信会限制你的时间的，30秒，发在朋友圈里的视频都是有时间限制

你也可以直接禁止这种行为

也可以用流的方式，一边读取大视频的字节流

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/142\_课程结束时给大家的一个忠告：一定要自己尝试去做/笔记.docx**

《142\_课程结束时给大家的一个忠告：一定要自己尝试去做》

带着大家撸出来最核心的一套架构，架构上的东西，很多东西 我觉得有方案，你自己可以做出来；还有很多东西对应的技术我之前都讲过，你完全可以自己去做一下；很多技术是比较简单的，查阅一些demo自己搭建一下；一些技术cdn，httpDNS，是云厂商提供的，查阅一下人家的产品使用方案

# **File Path: /Users/xiaotingting/Downloads/学习资料/13\_基于Netty的高并发基础架构项目实战unzip/124~143资料/132\_再看安全性：DNS劫持是什么意思？/笔记.docx**

《132\_再看安全性：DNS劫持是什么意思？》

域名 -> ip地址

如果你不是随便浏览不良网站，或者是钓鱼网站，中毒，中木马

家用路由器被黑了，dns被篡改了，指向其他非法分子的服务器，你访问百度，结果回来的是一个不良网站

[www.baidu.com](media/image3.png) -> 解析成一个ip地址，发送请求到百度的一台服务器上去

域名劫持

运营商LocalDNS被劫持，DNS缓存服务器，被劫持之后一样会这样

域名 -> 运营商的缓存LocalDNS服务器 -> 权威DNS服务器