**Gproxy设计文档**

**一、背景及介绍**

公司的第二代游戏引擎基于java语言实现，但是由于项目工期较赶，很多设计以简单

为主，游戏的整个通讯流程由客户端向服务器发起socket长连接，在自定义通讯协议基础上实现游戏网络通信。公司目前的主要游戏为mmorpg类型游戏，mmorpg类型游戏开发成本较高，开发周期较长，公司希望未来在一些更轻量级的游戏做一些尝试，游戏引擎需要率先开发出来。在此基础上，参考了php的网络通信框架workerman以及衍生出来的框架gatewayworker的一些设计，将客户端直连服务器的通信方式改成由客户端连接新的角色网关服务器建立socket连接，由网关服务器负责路由转发消息到游戏服，游戏服处理完再发送给网关服务器，网关服务器再将游戏服转发来的消息转发给客户端。

**二、技术选型**

选择java语言作为服务器引擎的技术栈，选择了netty作为通信框架，使用zookeeper

作为服务器地址注册及监听服务。

**三、架构设计**

4+1视图模型

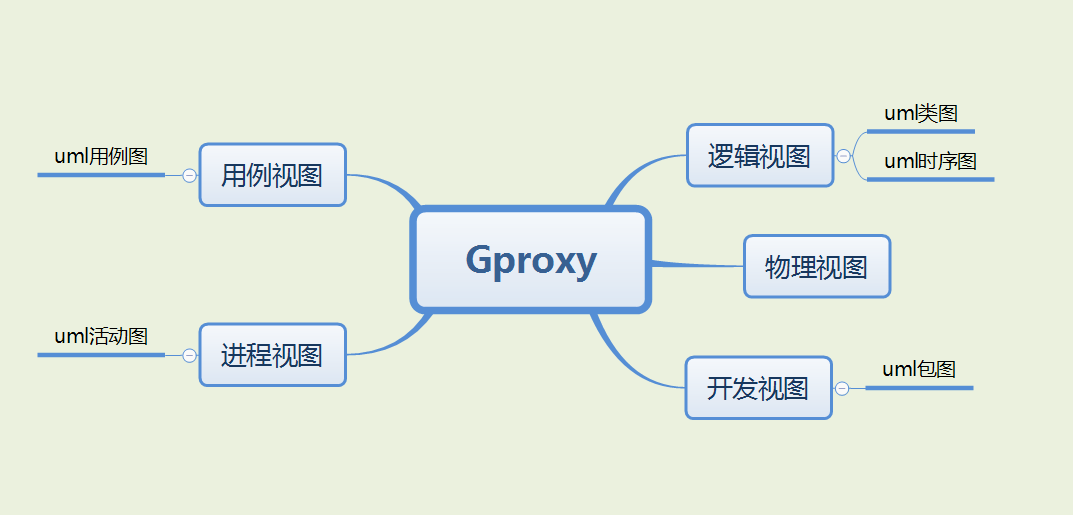
逻辑视图：主要用来描述系统的功能需求，在逻辑视图中，系统分解成一系列功能抽象、功能分解、功能分析

进程视图：主要考虑一些非功能性需求，它解决系统并发性、分布性、系统完整性、容错性等问题

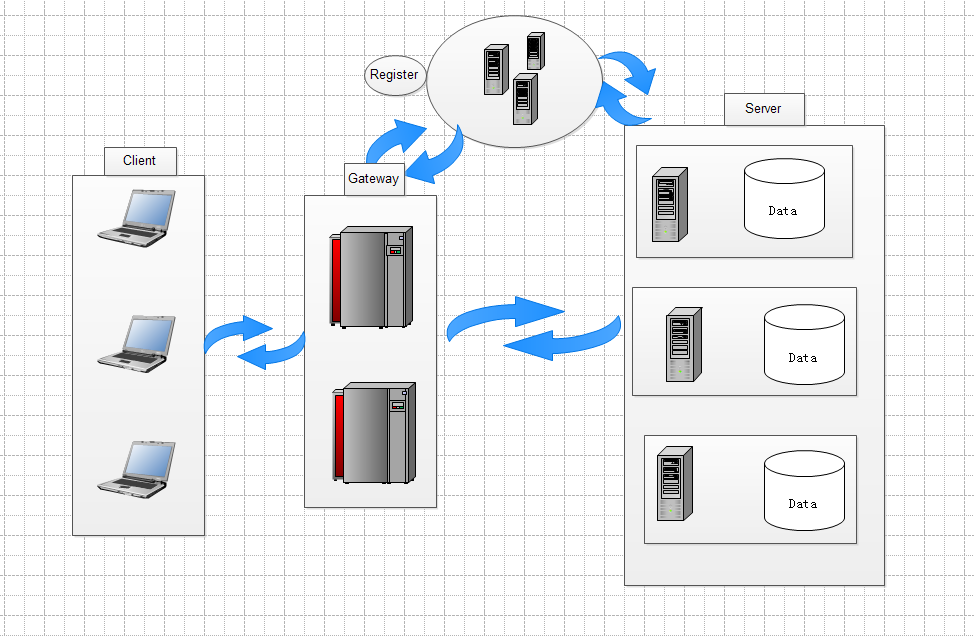
物理视图：主要描述硬件配置，服务于系统工程人员，解决系统的拓扑结构、系统安装、通信等问题

开发视图：主要描述了开发环境中软件的静态组织结构，服务于软件编程人员

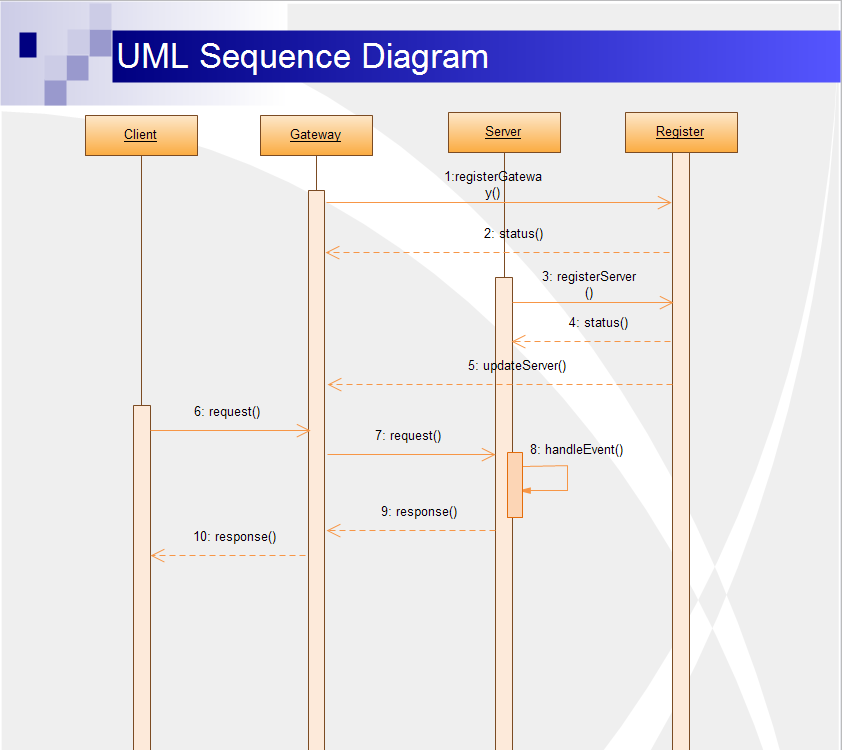
场景视图：主要描述一个特定的视图内的构件关系，也可以描述不同视图间的构件关系

架构视图 

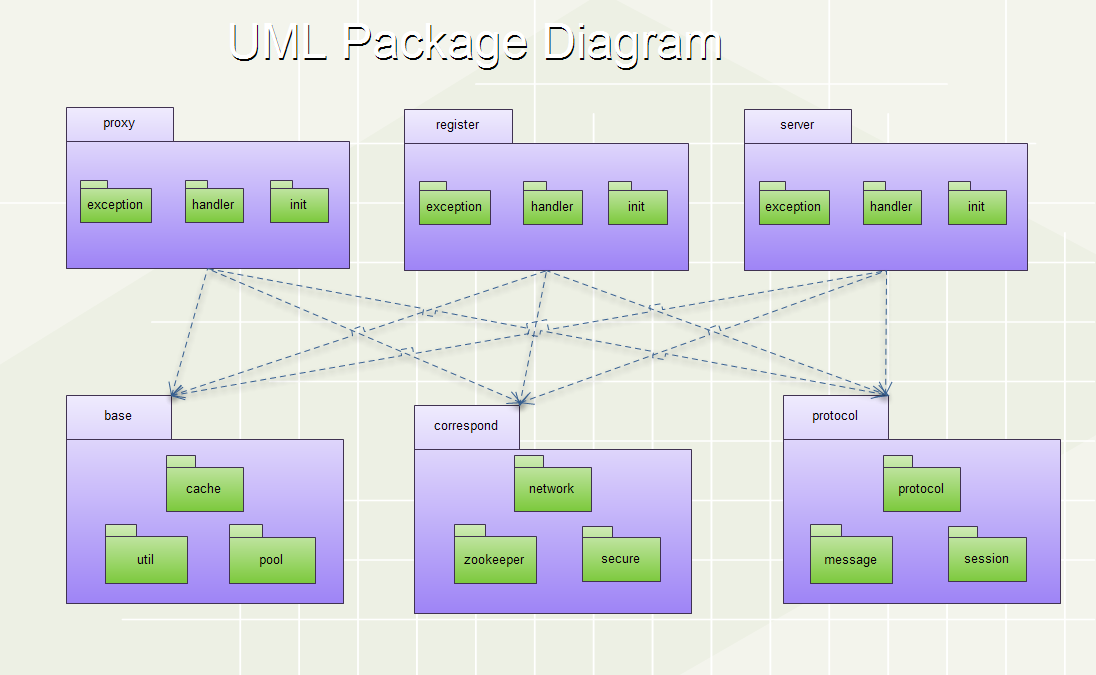
物理视图



时序图



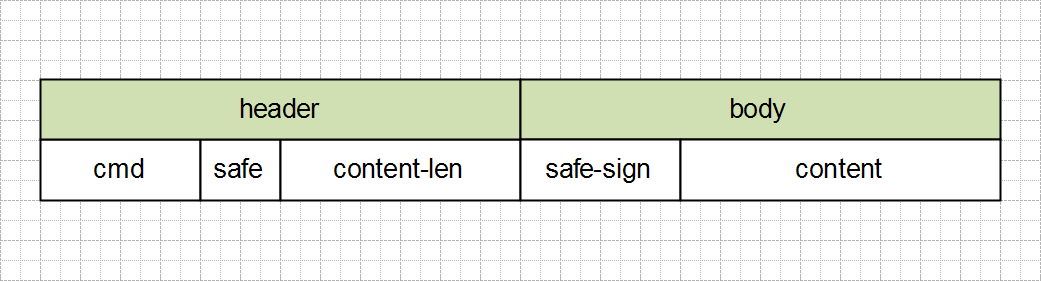
包图



**四、通信协议**

通信协议采用自定义二进制协议，最大化节约网络通信传输字节量。

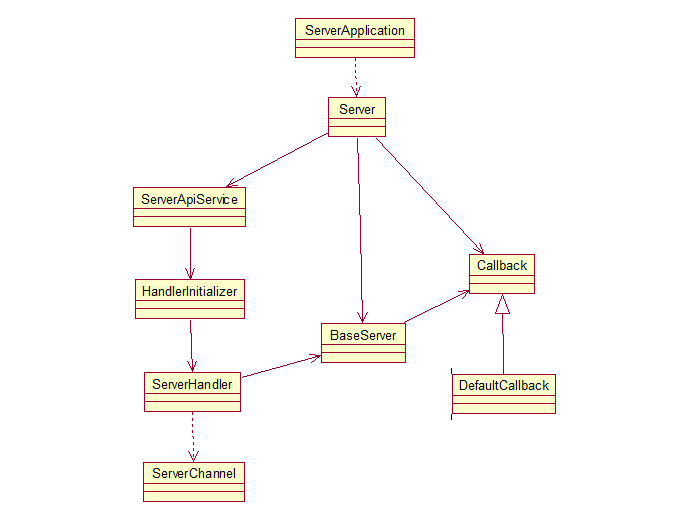
通信协议



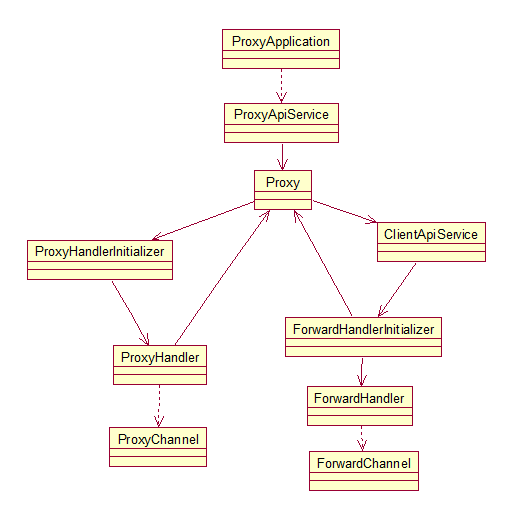
* cmd：8bit，可表示255种在协议命令
* safe：8bit，安全位，默认为0表示协议不加密，同时body中safe-sign不存在，如果为1，表示用safe-sign签名对content进行加密
* content-len：16bit，消息内容长度，消息体最长为2^17-1=131 071字节，如果safe安全位为1，需要分别读取safe-sign和content，然后用解密算法解密
* safe-sign：16byte，当safe安全位为1时，safe-sign才存在，content为被safe-sign加密过的内容
* content：长度未知，由content-len字段指定

**五、详细设计**

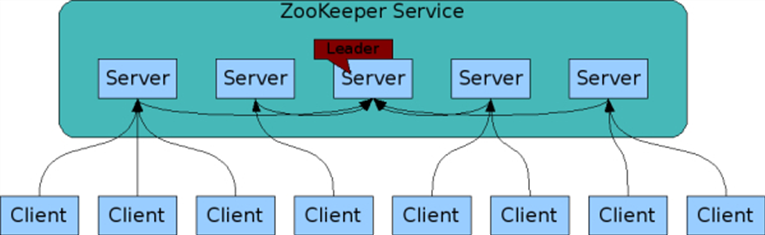
Proxy类图



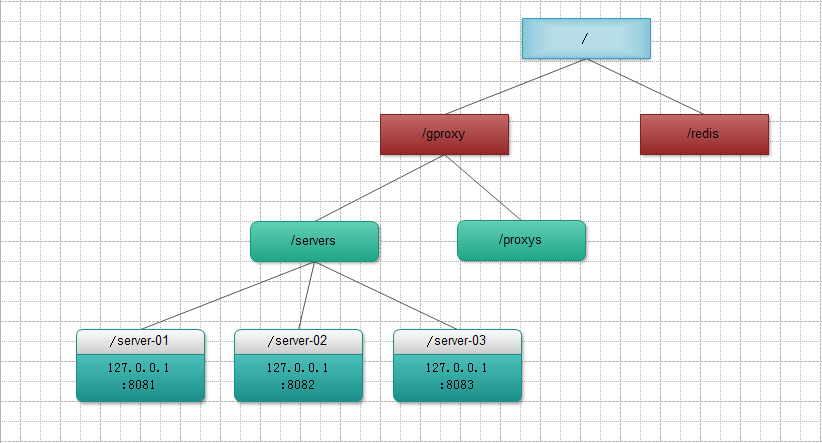
Server类图



zookeeper是一个开放源码的分布式应用协调服务框架，是hadoop和hbase的重要组件。它为分布式应用提供一致性服务，提供的功能包括：配置维护、域名服务、分布式同步、组服务集群。



zookeeper拥有一个层次化的命名空间，和标准的文件系统非常相似，可以往znode节点中存储数据或者从节点中获取数据。



**六、模块介绍**

Cache模块：主要提供多线程下系统内置KV缓存的存取

Client模块：主要是客户端测试

Cluster模块：主要提供zookeeper集群的负载均衡算法，实现了随机、轮询、一致性哈希负载均衡算法

Message模块：主要包含系统的常量类和对ip地址类的封装

Network模块：提供了netty的通道管理类，netty的socket客户端类和服务端类，是系统通信的基础

Pool模块：提供了多线程下系统线程池的封装，供二次开发使用

Protocol模块：主要包含了系统通信的协议实现类、对协议类的编解码、通信协议的命令定义以及一些其他协议处理辅助类

Proxy模块：主要包含了系统的网关类，包括接受客户端连接的处理、对客户端路由算法的处理，以及来自服务器消息的处理

Secure模块：主要提供协议加密与解密的包

Server模块：主要提供了服务端接受网关服务器的逻辑处理，抽象了二次开发的回调接口

Session模块：主要提供了系统上下文session相关类

Util模块：主要提供系统一些常用的工具类

Zookeeper模块：主要实现了对zookeeper的节点注册，节点获取，节点变动更新等方法的包装

User模块：主要提供开发者在回调接口上对消息进行二次逻辑开发

* public void sendToAll(String message); //向所有客户端发送该条消息
* public void sendToClient(String message, String clientId); //向指定clientId的客户端发送该条消息
* public void sendToUser(String message, String userId); //向指定userId的客户端组发送该条消息
* public void sendToGroup(String message, String groupId); //向指定groupId的客户端组发送该条消息
* public void bindUid(String clientId, String userId); //将clientId的客户端绑定到userId的用户组中
* public void unBindUid(String clientId, String userId); //将clientId的客户端从userId的用户组中解绑
* public void joinGroup(String clientId, String groupId); //将clientId的客户端绑定到groupId的组中
* public void leaveGroup(String clientId, String groupId); //将clientId的客户端从groupId的组中解绑