1. 概述

本文研究服务端架构

1. 应用场景

目前服务端采用UDP进行通信，同时也部分支持TCP方式。

1.对于UDP目前采用select单线程简单模型，优点是简单方便，避免线程切换，不用维护连接，对于简单请求响应良好。缺点是当处理复杂请求时，线程忙于计算，无法及时收取socket消息。好在我们的服务端逻辑处理比较简单。

客户端

客户端

客户端

服务端

线程

接收请求

应答请求

处理请求

2.为了解决以上问题，可采用select单线程收/发，多工作线程处理消息队列的模型

当客户端请求到来时，select线程收取消息，并放入消息队列，工作线程从消息队列取出消息，处理完发送反馈，或另用消息队列进行反馈。需要做好线程同步，对消息队列加锁。考虑使用无锁队列。

客户端

客户端

客户端

服务端

线程

接收请求

应答请求

加入请求消息队列

加入应答

消息队列

工作线程

取出

应答消息

取出

请求消息

线程

3.以上采用UDP传输，无法忽视可能的丢包问题。可部分采用TCP方式。每建立一个连接则发起一个线程维护此连接。优点是简单明了，缺点是连接过多时线程数过多，导致线程间切换开销大。一种优化方法是采用select + 线程池

客户端

客户端

客户端

服务端主线程

accept

工作线程

接收请求

应答请求

创建线程

工作线程

工作线程

4.满足高并发要求的服务器，可以采用iocp（win）或epoll（linux）