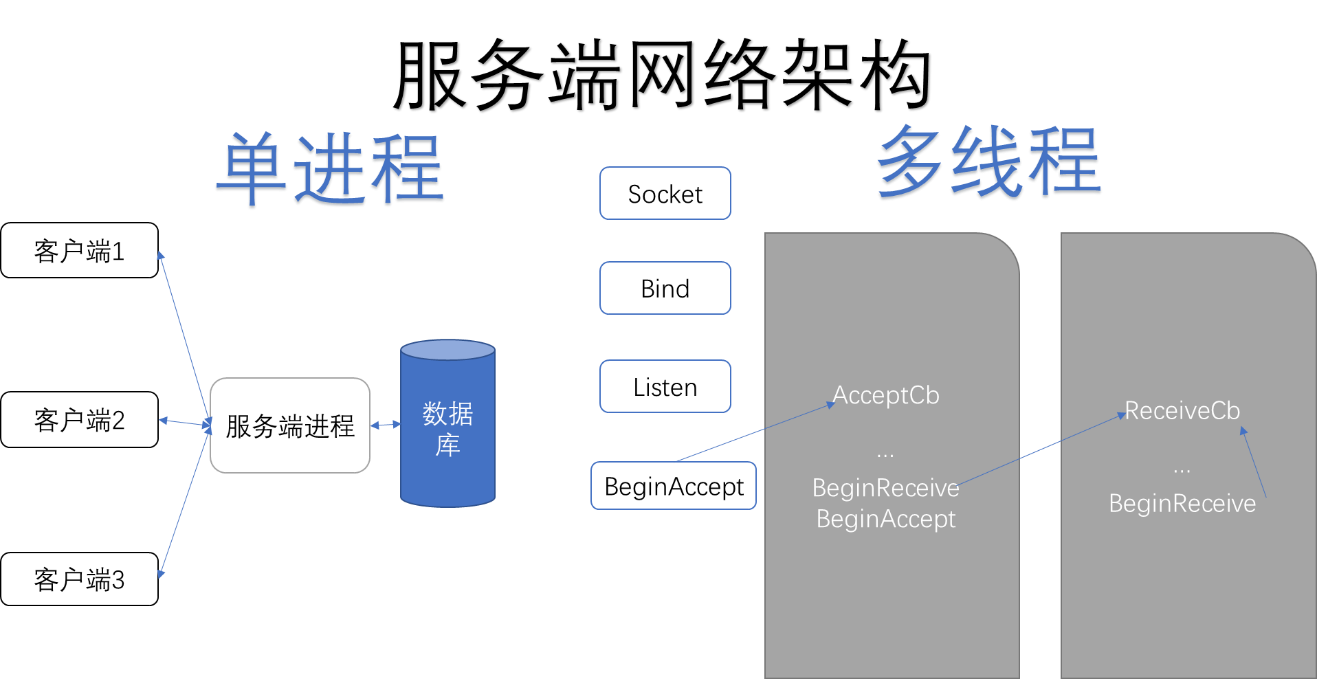
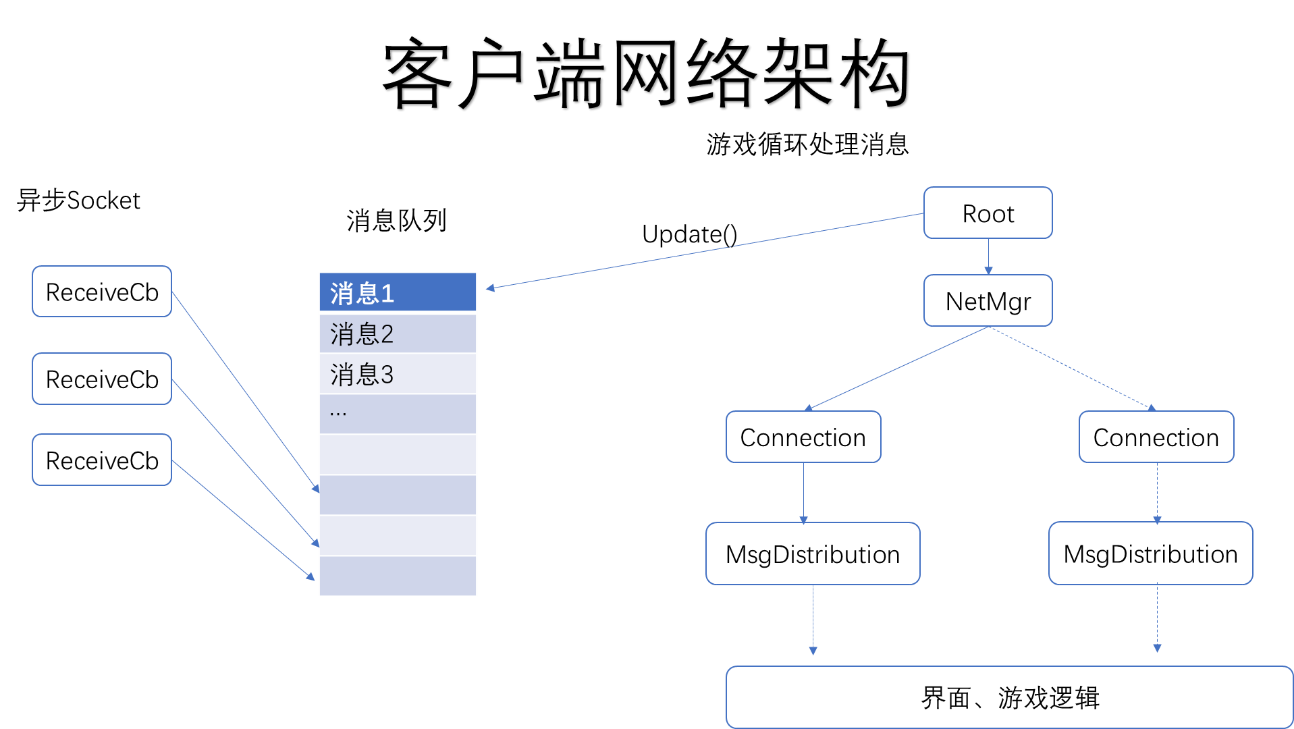
# 消息处理架构

服务端采用的是单进程多线程的异步消息处理方式，客户端多线程的异步接受消息，单线程地消息处理（很多处理函数和组件只能被主线程使用，并且为了保证客户端游戏逻辑时序性，我们采用单线程的消息处理）

## 服务端网络架构



## 客户端网络架构



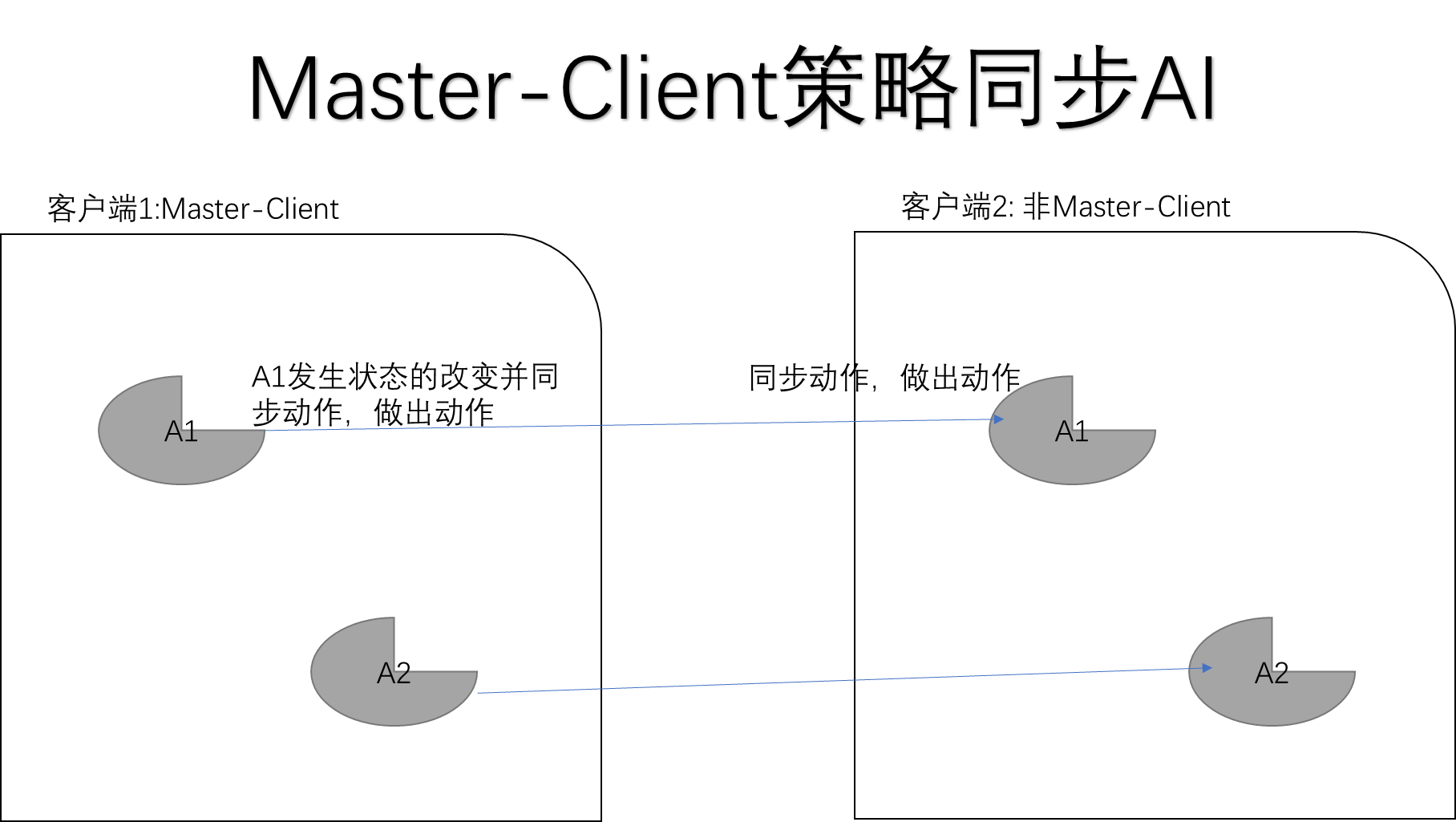
## 数据库设计

user表：存放用户的账户信息

player表：存放角色信息

**由于角色信息可能包含具有复杂层次结构数据类型的角色数据，故采用类的序列化来存储角色信息**

## Master-Client策略同步AI

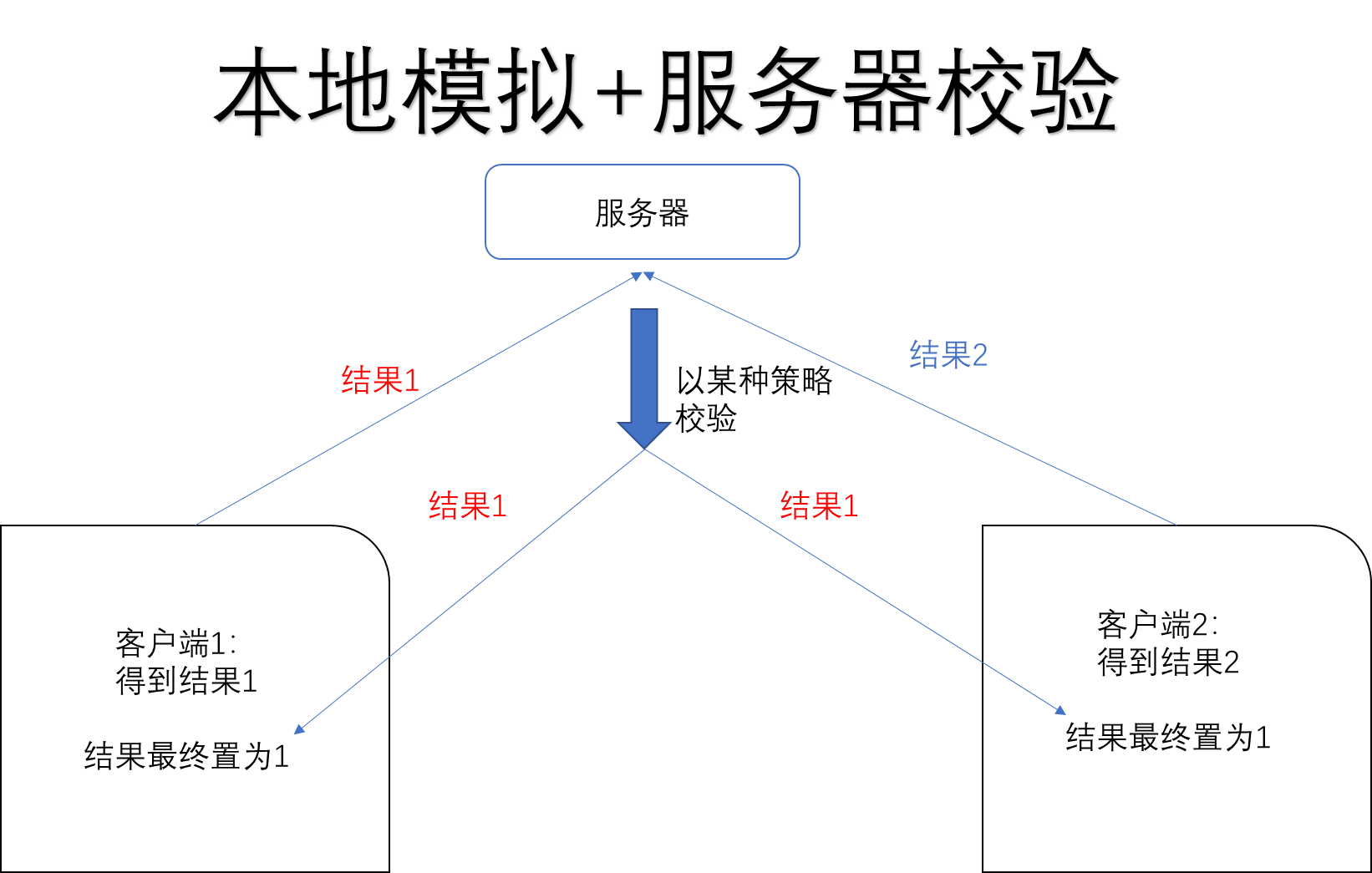


# 客户端计算结果的状态同步架构

协议之间传输的是游戏状态的结果。

* 玩家的位置与AI的位置的同步采取了航位推算原理的预测机制。每个实体5HZ的更新频率即可保证位置同步的流畅性。
* AI的同步采取了master-client的方式来保证一致性。只有一个客户端拥有完整AI客户机脚本，该客户端的AI来操纵其他客户端的AI的行为。
* 玩家事件（射击、触发战场事件）为保证流畅性采取了本地先模拟结果同时同步结果至其他客户端的策略。

## 本地模拟+服务器校验



为了增加流畅性，采取本地先预测得到结果同时同步到其他客户端的策略。

### 优点

迅速响应结果，不用等待RTT时间才响应。

### 缺点

发生在几十毫秒内的事件可能会出现短暂的不同步情况，我们允许这种情况的存在，但在事后采取校验的方式恢复一致。

# 通用的网络同步组件

* 网络同步组件挂载在需要同步的GameObject上便可便捷地调用网络接口。
* 提供同步变量和同步函数调用的接口
* 大大降低了团队人员使用网络模块的难度。

