第33卷 第15期 计 算 机 工 程 2007年8月
Vol.33 No.15 Computer Engineering August 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)15—0283—03 文献标识码：A 中图分类号：TP393

棋牌类网络游戏服务端的架构设计

吴兆定，袁江海，郑世宝
（上海交通大学图像通信与信息处理研究所，上海 200240）

摘 要：网络游戏目前在国内相当热门，棋牌类游戏在网络游戏中占有比较重要的地位。越来越多的开发商加入到了网络游戏的开发中来，但有关网络游戏开发技术介绍的资料却比较少。该文总结了一款通用的棋牌类网络游戏服务端的架构设计，介绍了基本架构、通信协议、多线程模型和第3方接口的相关内容，并对整个架构作了一定分析。

关键词：网络游戏；服务器；服务端架构；通信协议

Design of Server Architecture for Chess/Cards Online Games

WU Zhao-ding, YUAN Jiang-hai, ZHENG Shi-bao
(Institute of Image Communication & Information Processing, Shanghai Jiaotong University, Shanghai 200240)

【Abstract】 Online games are very popular now in China, among which the chess/cards online game is one of the hottest. But there are few papers about how to develop an online game. Based on the work on the practical project, this paper presents a universal server architecture for chess/cards online games. The architecture, communication protocol, multi-thread model and interface for third-party of this server are discussed. It analyzes the overall architecture of the game server.

【Key words】 online game; server; server architecture; communication protocol

棋牌类休闲游戏是网络游戏中非常重要的一类。从早期的联众到现在的QQ游戏，棋牌类休闲游戏得到了长足的发展。由于这类游戏是将传统的民间游戏网络化，玩家很容易上手，因此这类游戏成了受众群最大的网络游戏之一。

作为研究开发人员，比较关心如何去开发这样的游戏。然而，由于商业竞争的原因，目前网络上、期刊上很难找到相关的技术文档。本文结合工程开发实践提出了一种通用的棋牌类网络游戏的服务端架构，希望对读者能有一定的参考价值。

1 问题

在设计这个通用棋牌类网络游戏的服务端架构时，必须考虑以下4个问题：

(1)高效性：网络游戏服务端的首要目标就是能为大量的客户端服务，同时还应该具备相当的游戏逻辑处理速度。

(2)安全性：服务端要求能够对一些恶意攻击有一定的防护能力。

(3)扩展性：一个好的网络游戏服务端架构，要求在一定程度上，能动态地增加或者减少当前的服务端规模。

(4)通用性：对于一个通用的棋牌类游戏平台来讲，其通用性就显得相当重要。平台要求能兼容第3方开发的游戏。

下面就以上面提出的4个问题为目标，给出一种通用的棋牌类网络游戏的服务端架构的设计。

2 设计

2.1 架构

棋牌类游戏与其他网游最大的不同点在于，其游戏逻辑比较简单，通常一台服务器就可以承担一种甚至多种游戏的逻辑处理，而其他游戏却不一定是这样。比如典型的MMORPG游戏，一般一台服务器只负责一块地图的逻辑处理，还可能有NPC服务器、物品掉落服务器等专门服务器。另外，棋牌类游戏的结构相对而言也比较固定和单一，一般由大厅和游戏房间组成，大厅是选择游戏的平台，玩家在房间里进行游戏。针对这些特点，给出了如下一种架构：

一种通用的棋牌类网络游戏服务端的架构如图1所示，主要分为5种软服务器：登录服务器(login server, LS)，大厅服务器(hall server, HS)，中央服务器(main server, MS)，游戏服务器(game server, GS)和数据库服务器(DB)。所谓“软服务器”是指逻辑意义上的服务器，并不等于现实中的一台具体的物理服务器。一台物理服务器上可以运行多个软服务器，一个软服务器也可能是由多台物理服务器所组成。

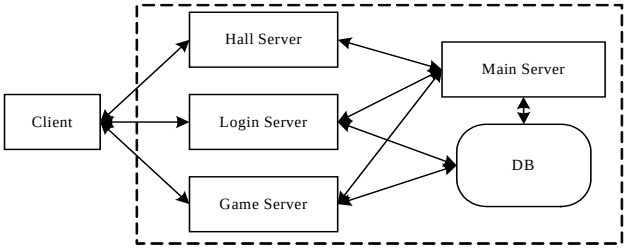

图1 服务端架构示意图

LS用于对玩家进行登录验证。玩家通过验证后，就与LS断开连接，以腾出Socket资源。根据玩家数量的多少，可以设置多个LS，这时可采用一定的负载均衡策略，如最简单的基于DNS的负载均衡。

玩家通过LS验证之后，就会与HS发起连接。一旦客户端和HS建立起合法连接，玩家就进入了游戏的大厅。此后玩家一直与HS保持连接，这样可以很方便地检测玩家的意

作者简介：吴兆定(1981－)，男，硕士研究生，主研方向：网游服务端设计；袁江海，助理研究员；郑世宝，教授、博士生导师
收稿日期：2006-08-27 **E-mail：**solve@sjtu.edu.cn

外断线等情况。HS 也可以根据负载设置多个，负载均衡算法下面给出例子说明。

MS 可以说是整个服务端的核心，它用于存储一些全局信息，如厅中玩家信息、游戏服务器的信息等等。MS 还负责一些合法性检验和消息的转发等功能。为保证系统的安全性，整个过程中玩家不会直接与 MS 进行连接和交互。

GS 在客户端看来，就是提供游戏的一个房间。玩家在客户端点击进入某个房间，就会与相应的 GS 建立连接，玩家离开 GS 时便与之断开连接。

DB 用于存储玩家的游戏信息，包括账号信息和游戏中的数据等等。为保证系统安全，玩家不会直接与 DB 进行连接和交互。

以上介绍了几个软服务器的功能，下面给出一个具体的玩家从登录到退出的例子，说明服务端如何实现相应功能。玩家首先通过某种负载均衡算法选择一个 LS 并发出登录消息。LS 收到消息后，到 DB 查询账号的合法性，如果合法，则向 MS 发送验证重复登录的消息。若 MS 上没有此玩家已登录的信息，则在 MS 上的玩家列表中加入该玩家，同时 MS 给 LS 返回一个 HS 的连接信息。HS 的负载信息在 MS 上有记录，这样 MS 就可以选择一个负载最轻的 HS 告知客户端。客户端收到后便向 HS 发出连接请求，HS 向 MS 发送该客户端的合法性验证请求。如果通过验证，MS 将通过 HS 将当前各游戏服务器的信息传送给客户端，方便玩家选择进入哪个游戏房间。经过上述步骤，玩家便进入了大厅。接下来，玩家会选择进入某个游戏房间。同样，玩家必须通过 MS 的合法性验证才能与 GS 建立连接。建立连接后，房间中的所有逻辑都由 GS 处理。GS 最重要的功能就是负责具体游戏的逻辑处理，为保证平台的通用性，设计了一套第 3 方接口，这样游戏逻辑就可由第 3 方根据接口来开发，将在 2.4 节具体讨论。玩家退出房间时，GS 进行相应的数据保存后与客户端断开连接即可。玩家退出整个游戏时，与 HS 断开连接，HS 告知 MS，MS 删除玩家信息，玩家即成功退出游戏。

2.2 通信协议

网络游戏一个重要的特点就是存在网络通信，如客户端和服务端的通信，还有服务端之间的通信。为保证系统的可靠性，所有的通信都采取了 TCP 协议。由于网络游戏的用户量一般都非常大，因此在设计客户端和服务端之间的通信协议时，采用了完成端口(Completion Port)的技术，大家可参考相关资料。

下面重点介绍一下软服务器之间的通信协议。我们没有采用现成的网络中间件，如 DCOM、CORBA 等。一方面是为了节约开发成本，另一方面这些网络中间件的诸多功能也用不上。根据自己的需求设计了一套通信协议，称为星形分布式通信协议(Star Distributed Communication Protocol, SDCP)。由图 1 可以看出，整个服务端(不包括 DB)的架构是一个星型结构，MS 是中心节点(Center Node, CN)，而其他服务器是叶节点(Leaf Node, LN)。这里没把 DB 考虑在内，是因为一般 DB 本身都会提供相应的通信功能。实际上，SDCP 仍然是一种 C/S 模式的通信协议，CN 作为 Server，而 LN 作为 Client。为 CN 设计了一个 CenterNetbase 类，为 LN 设计了一个 LeafNetbase 类，片段代码如下：

```
class CenterNetbase
{public:   //开始监听
bool StartListen(int listenPORT);
Message* GetMsg(); //获取 LN 的消息
//向 LN 发送消息
void SendMsg(Message* outData);
…};
class LeafNetbase
{public:
//向 CN 发起连接
bool Connect(ConnectInfo *target);
bool DisConnect();   //与 CN 断开连接
Message* GetMsg(); //获取 CN 的消息
//向 CN 发送消息
void SendMsg(Message* outData);
…};
```

在 CN 首先启动好的前提下，SDCP 允许 LN 随时向 CN 发起连接请求，这样就实现了整个系统的可扩展性。当觉得 LS、HS 或者 GS 的负载过重时，完全可以动态地增加相应的服务器。各服务器之间的 TCP 连接，也保证了服务器之间通信的高可靠性要求。

上面曾提到，几个软服务器可以设置在同一台物理服务器上。SDCP 中的节点是以 socket 来区分的，因而，对同一台物理服务器上的软服务器，只需要为它们分配不同的端口就能满足 SDCP 的要求了。服务器之间的通信都是通过消息传递来实现的，对消息的格式定义如下：

```
class Message
{public:
int size;          //消息的字节数
int senderID;   //消息发送方的标识
int type;          //消息类型
char* pMsg;     //指向消息的首字节
…};
```

消息的结构如图 2 所示。Message 类中还提供了一系列用于构造消息和解析消息的方法，主要用于将需要发送的信息拼装成一条完整的消息，接收方收到后再根据消息类型解析出所包含的信息。这里就不一一介绍了。

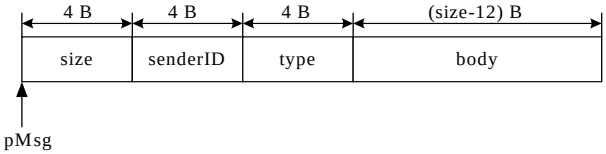

图 2 消息结构

2.3 多线程模型

对玩家的操作响应越快，服务器的性能就越好。为了达到这个要求，最好能为每个玩家在服务端启动单独的线程来处理消息。但这种做法是不现实的。一方面，操作系统的最大并发线程数限制了最大用户量；另一方面，如果采用多线程处理，那么玩家的数据同步问题就会显得相当复杂。经权衡，设计了这样一个多线程模型。

以 MS 为例，MS 的 CenterNetbase 包含两个缓冲队列，inQueue 和 outQueue。inQueue 用于存放接收到的消息，outQueue 用于存放待发送的消息。CenterNetbase 中将启动两个线程，接收线程和发送线程。接收线程不停地将网络上接收到的消息存放到 inQueue 中，而发送线程不停地将 outQueue 中的消息发送给相应接收者。MS 还将启动唯一的一个主线程，用于处理 inQueue 中的消息。在处理过程中，当需要往外发送消息时，则将消息插入 outQueue。整个多线程模型如图 3 所示。

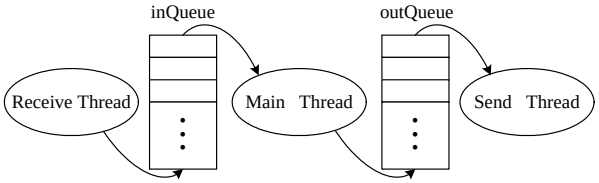

图 3　MS 的多线程模型

LS、GS 和 HS 的多线程模型和 MS 有点不一样。这 3 种服务器除了含有类似于 CenterNetbase 的 LeafNetbase 之外，还有一个 ServerNetbase 类用于处理和客户端的通信。同样，ServerNetbase 中也包含两个线程和两个队列。为了保证数据同步处理的简单，在主线程中仍然只启动唯一的一个线程，只不过这时需要处理两个 inQueue 中的消息。扩展一下，当 Netbase 类不止是两个时，就得到了一个可复用的多线程模型。处理线程的片段代码如下：

```
while(1){
//对 inQueueCount 个 inQueue 依次处理
for (int i=0; i<inQueueCount; i++){
msgCount = inQueue[i].size();
//将此 inQueue 中的当前消息处理完
for (int j=0; j<msgCount; j++){
//获取消息
msg = netbase[i]->GetMsg();
if (msg != NULL){
HandleMessage(msg); //处理消息 } } }
Sleep(interval); }
```

2.4　第 3 方接口

由于不同游戏的逻辑不一样，如果将游戏逻辑与 GS 绑定的话，那么不同的游戏将具有不同的 GS，这样大大影响了平台的通用性。我们希望在这个平台上，能比较方便地运行多种游戏，因而设计了一套第 3 方接口，来兼容第 3 方开发的游戏逻辑处理程序。

采取了动态链接库(dll)的形式来实现第 3 方接口。对于 GS，如果它们承担不同的游戏，只需要动态加载不同的游戏 dll 就可以了。为尽量简化第 3 方接口，我们通过消息机制来完成 GS 和第 3 方 dll 之间的通信。接口类定义如下：

```
class ThirdInterface
{public:
bool Initialize();
bool Stop();
void HandleThirdMsg(ThirdMsg* msg);
ThirdMsg* GetGameMsg();
…};
```

接口类存在于 dll 中，并可动态导出，以供 GS 使用。Initialize 和 Stop 分别用于初始化和关闭第 3 方程序。在客户端，也相应地提供了一套第 3 方接口。第 3 方服务端会和第 3 方客户端进行消息通信，这些由平台实现。将这些消息定义成一类特殊的消息，当服务端接收到这样的消息时，就调用第 3 方接口 HandleThirdMsg，让第 3 方来对这些消息进行处理。当第 3 方服务端要求发送消息时，就将消息放到一个缓冲队列中去，GS 的主线程会不断地调用接口函数 GetGameMsg 来获取缓冲队列中的消息，发现是需要往客户端发送的消息时，就将其插入到 outQueue 中。GS 和第 3 方服务端之间的其他信息传递也通过以上方式进行。

2.5　其他

在设计开发过程中还有很多细节性的东西需要注意，这里简单列举一二。比如，一般情况下 LS、HS 和 GS 都是几个运行在同一台物理服务器上，而 MS 则需要由一台甚至多台物理服务器来实现。当 MS 成为系统瓶颈时，就需要将 MS 设计成一个单接口的应用集群服务器。还比如，为了进一步减轻服务端的负载，可以将非关键逻辑消息直接在客户端之间传送，而不通过服务端。但这是在不影响游戏安全性的前提下的改进，这就要求进一步去区分关键逻辑和非关键逻辑。再比如，可以为所有的消息设定一定的优先级，进一步提高系统的性能。

3　分析

以上就是给出的设计方案，现在来回头看一下第 1 节中所提出的 4 个问题。

(1)高效性：在设计系统结构时提出了软服务器的概念，多个软服务器可以运行在同一台物理服务器上，这样可以灵活地根据负载来搭配服务器，能更好地利用服务器资源。LS 和 HS 都有一定的负载均衡算法，不会出现某个服务器负载过重导致崩溃的情况。多线程模型也保证了数据操作的便捷性。另外，开发时使用的是 C++这种效率比较高的编程语言，在开发过程中也充分利用了它的高效性。比如接收消息线程从网络上接收到消息以后，通过 C++强大的内存管理能力，就能直接对消息所在的内存进行一系列操作，而不用再将消息中的信息再次复制出来使用，尽量减小内存拷贝的次数，也极大地提高了系统的性能。

(2)安全性：将最核心的 MS 没有暴露给玩家，这大大提高了系统的安全性。另外，在设计开发过程中，充分考虑了玩家的合法性验证，比如在玩家登录 LS、HS 和 GS 时，都做了相应的合法性检验，这也提高了系统的安全性。当然，还可以对某些重要的通信消息进行加密，进一步提高系统的安全性。

(3)扩展性：把架构设计成星型结构，并提出了 SDCP 通信协议。在 CN 已经启动的前提下，通过 SDCP，能非常方便地对 LN 进行动态扩展。这使得在系统中 LS、HS 和 GS 都有非常大的扩展空间，能承受更多的用户量。

(4)通用性：为了实现平台的通用性，为客户端和服务端分别定义了一套第 3 方接口(由于篇幅原因，只介绍了服务端的接口)。如果第 3 方开发商能按照这套接口去开发游戏逻辑，那么这些游戏就能很好地集成到平台中来。在实际开发过程中，已经开发了两款不同的游戏，这两种游戏按照第 3 方接口的标准开发，通过 DLL 形式很好地跟平台集成了起来。

4　小结

本文提出的服务端的系统架构，在一定程度上比较好地解决了高效性、安全性、扩展性和通用性的问题。当然，本系统架构仍然存在很多不完善的地方，希望与读者进行交流，能够提出更好的解决方案来与大家分享，让网络游戏开发者都得到进一步的提高。

参考文献

1 Ohlund J. Windows 网络编程[M]. 杨合庆, 译. 北京: 清华大学出版社, 2002.

2 苏　羽, 王媛媛. Visual C++网络游戏建模与实现[M]. 北京: 北京科海电子出版社, 2003.

3 李文正, 郭　巧, 王　利. Internet 服务器负载均衡的研究与实现[J]. 计算机工程, 2005, 31(6): 98-99.

4 唐　磊, 金连甫. 一种新的大型通用分布式服务器架构[J]. 计算机工程与设计, 2004, 25(10): 1784-1786.