重庆大学本科学生毕业设计（论文）

网络五子棋程序的设计与实现

学　　生：陈磊

学　　号：20075365

指导教师：刘骥

专　　业：计算机科学与技术

重庆大学计算机学院

二 O 一一年六月

Graduation Design(Thesis) of Chongqing University

Design and Implementation of Network Gobang

Undergraduate: Chen Lei

Supervisor: Lecturer Liu Ji

Major: Computer Science and Technology

College of Computer Science

Chongqing University

June 2011

摘　要

在现在如此发达并且得到广泛应用的现代网络技术下，集趣味性，娱乐性，益智性，并且包含网络功能的休闲类游戏以及成为了人们选择网络休闲游戏的要求。

系统采用当今广为流行的五子棋游戏为模版，利用 C++的第三方 GUI 设计工具 Qt 为程序设计界面，并结合软件工程的思想开发一款基于网络的五子棋游戏对弈软件。本软件采用 P2P 的模式，利用一个服务端来辅助各个客户端之间的查找和链接，服务端用于记录当前各个客户端的游戏状态，并将实时更新的各个客户端的状态发送到其他客户端，方便客户端加入其他客户端建立的游戏主机，并建立 P2P 链接，进行游戏。客户端是玩家用于建立游戏主机或者加入游戏进行五子棋对弈的主程序，具有当今五子棋游戏常见的聊天功能、悔棋功能、认输功能、计分功能、玩家断网处理等。

玩家只需运行客户端，连接到指定的服务端，就可以加入其他的游戏主机进行游戏，或者建立游戏主机，等待其他客户端的加入进行游戏对弈。

关键字：多线程，Qt，socket

ABSTRACT

With the development and widely usage of computer network, it has been a requirement for a casual online game to be interesting, entertaining and intelligence-improving.

This paper implements a online Gobang game based on the popular Gobang, used third-party C++ GUI library Qt and combined software engineering model. The software uses P2P model, which uses a sever end to help clients find and connect with each other and help record the status of every client and send it to other clients in order to make it convenient to join the host and establish the P2P connection between clients to start the game. Client is the main program for user to create or join the Gobang game and have the functionalities of chatting, regretting, yielding, scoring and connection problems handling.

All user needs to do is run a client, connect to the specific server, then he can join a host or create a host himself and wait for other clients to join and have a game.

Key words： multithreading, Qt, socket

目　录

1 绪论

1.1 课题研究意义

五子棋[1]是起源于中国古代的传统黑白棋种之一。现代五子棋日文称之为“连珠”，英文称之为“Renju”、“Gobang”或“FIR”（Five in a Row 的缩写），亦有“ 连五子”、“五子连”、“串珠” 、“五目”、“五目碰”、“五格”等多种称谓。

五子棋不仅能增强思维能力，提高智力，而且富含哲理，有助于修身养性。五子棋既有现代休闲的明显特征“短、平、快”，又有古典哲学的高深学问“阴阳易理”；它既有简单易学的特性，为广大人民群众所喜闻乐见，又有深奥的技巧和高水平的国际性比赛；它的棋文化源渊流长，具有东方的神秘和西方的直观；既有“场”的概念，亦有“点”的连接。它是中西文化的交流点，是古今哲理的结晶。

五子棋起源于古代中国，发展于日本，风靡于欧洲。对于它与围棋的关系有两种说法，一说早于围棋，早在“尧造围棋”之前，民间就已有五子棋游戏；一说源于围棋，是围棋发展的一个分支。在中国的文化里，倍受人们的青睐。古代的五子棋的棋具与围棋相同。五子棋大约随围棋一起在我国南北朝时先后传入朝鲜、日本等地。从此，五子棋经过了不断的改良，例如棋盘由原来的纵横各十七道改为现行的纵横各十五道的五子棋专用棋盘等等。二十世纪初，五子棋传入欧洲并迅速风靡了全欧洲。通过一系列的变化，使五子棋这一简单的游戏系统化、规范化，最终成为今天的职业连珠五子棋，同时也成为一种国际比赛棋。

在网络发达的今天，对于二人对弈的五子棋，开发一款带有网络功能的五子棋游戏，能够更加方便身在异地的玩家进行游戏。

1.2 课题研究目的

五子棋在我国的历史可谓是源远流长，喜爱它的玩家也是大有人在。但目前的五子棋软件一般都是单机版的，游戏无法在异地的玩家之间进行，因此为了方便异地的玩家在一起进行五子棋游戏，设计出一款基于网络版的五子棋游戏，玩家可以通过网络挑选自己喜爱的对手并同其进行比赛，比赛过程中可以进行聊天，达到相互交流经验的作用，同时能让身处各地的玩家连接到服务器进行五子棋对弈。

1.3 研究现状分析

目前常见的单机类五子棋游戏可分为“人机模式”与“双人模式”。“人机模

式”指人与按照事先编写好的五子棋落子算法程序来进行游戏的计算机进行对弈，“双人模式”指两个人通过使用同一台计算机，交替落子来进行游戏。这两种方式，都需要玩家双方在同一台计算机面前进行完成，因此局限性很大，玩家不能再异地的不同的计算机上来进行游戏。

随着现代计算机网络的迅速发展，网络上的计算机之间的通信已经变得十分便捷。这也为身在异地间的用户利用网络来进行游戏提供了途径。

网络版的五子棋软件利用网络连接协议 TCP/IP，在不同地方的玩家之间建立一个 TCP 连接，并用这个连接来交换数据，进行通信。因此，网络版的五子棋软件在游戏之前必须确定对网络参数设置的正确性，设置正确之后，才能进行玩家双方的 TCP 成功连接。连接成功后，双方轮流下棋，同时将每一步落子的信息通过网络连接传送给对方，使双方棋盘上的状态保持一致。

与单机版的五子棋软件相比，网络版的五子棋软件也提供了所有单机版的五子棋软件所具有的功能，例如：倒计时器、“认输”、“和棋”和“悔棋”等。但对于网络版的五子棋来说，这些功能的同步都是通过网络信息交互来实现的，因此，网络版的五子棋游戏实现过程较单机版要复杂一些。

在胜负判断方面，单机版的五子棋软件与网络版的五子棋软件可以采用完全相同的算法（如果不考虑先手禁手等）。一方玩家下完一步棋后，系统会自动判断当前落子之后，玩家是否胜利，如果没有胜利，再将落子的信息传送给对方，如果判断结果为胜利，则直接发送胜利消息给对方。每次落子之后，都要进行这样一次胜负判断定的过程。另外，由于网络版的五子棋的游戏双方可能不在同一个地点，因此，提供聊天功能很有必要。

目前常见的网络五子棋的游戏形式又可分为两种，一种是以 QQ 五子棋（如图 1.1）为代表的桌面应用程序的游戏形式，一种是网页 flash（如图 1.2）的游戏形式。

图 1.1 QQ 五子棋游戏

图 1.2 网页 flash 五子棋

系统结构方面，QQ 五子棋游戏采用了 C/S 结构，而网页 flash 的五子棋采用的是 B/S 结构。

功能方面，二者均提供了常见的悔棋、计分、计时、聊天、认输等功能。均未设计五子棋中“先手禁手”等规则。不同之处在于，常见的网页五子棋对于一方断网的情况不能及时发现，发现后也只是做出提醒另外一方的操作。而 QQ 五子棋能够对一方断网的情况及时发现，并且采用电脑托管或者直接结束比赛并计算相应分数的操作。

游戏流畅度方面，网页 flash 的游戏形式受限于浏览器解析网页代码的速度限制，游戏载入时间明显过长。而 QQ 五子棋只需要基本的 windows 操作系统即可，运行环境要求较低，运行流畅。

1.4 各章内容简介

本文第一章介绍了课题的研究背景，第二章简单介绍了实现程序所用到的技术介绍，第三章对程序需求进行了分析，第四章为系统的设计思想以及详细设计，第五章介绍了系统的实现过程，第六章展示了系统的效果，最后第七章总结了系统已完成的工作和下一步待完成的工作。

2 实现技术

2.1 P2P 模式

2.1.1 P2P 模式介绍

P2P[2]（peer-to-peer， 简称 P2P）又称对等互联网络技术，是一种网络新技术，依赖网络中参与者的计算能力和带宽，而不是把依赖都聚集在较少的几台服务器上。请注意与 point-to-point 之间的区别，peer-to-peer 一般译为端对端或者群对群，指对等网中的节点；point-to-point 一般译为点对点，对应于普通网络节点。P2P 网络通常用于通过 Ad Hoc 连接来连接节点。这类网络可以用于多种用途，各种文件共享软件已经得到了广泛的使用。P2P 技术也被使用在类似 VoIP 等实时媒体业务的数据通信中。

纯点对点网络没有客户端或服务器的概念，只有平等的同级节点，同时对网络上的其它节点充当客户端和服务器。这种网络设计模型不同于客户端-服务器模型，在客户端-服务器模型中通信通常来往于一个中央服务器。

2.1.2 P2P模式的优势

P2P网络的一个重要的目标就是让所有的客户端都能提供资源，包括带宽，存储空间和计算能力。因此，当有节点加入且对系统请求增多，整个系统的容量也增大。这是具有一组固定服务器的Client-Server结构不能实现的，因为在上述这种结构中，客户端的增加意味着所有用户更慢的数据传输。P2P网络的分布特性通过在多节点上复制数据，也增加了防故障的健壮性，并且在纯P2P网络中，节点不需要依靠一个中心索引服务器来发现数据。在后一种情况下，系统也不会出现单点崩溃。

当用P2P来描述Napster 网络时，对等协议被认为是重要的，但是，实际中，Napster 网络取得的成就是对等节点（就像网络的末枝）联合一个中心索引来实现。这可以使它能快速并且高效的定位可用的内容。对等协议只是一种通用的方法来实现这一点。

2.1.3 P2P模式的应用

点对点技术有许多应用。共享包含各种格式音频，视频，数据等的文件是非常普遍的，实时数据（如 IP 电话通信）也可以使用 P2P 技术来传送。

有些网络和通信渠道，像 Napster，OpenNAP，和 IRC@find，一方面使用了主从式架构结构来处理一些任务（如搜索功能），另一方面又同时使用 P2P 结构来处理其他任务。而有些网络，如 Gnutella 和 Freenet，使用 P2P 结构来处理所有的任务，有时被认为是真正的 P2P 网络。尽管 Gnutella 也使用了目录服务器来方便

节点得到其它节点的网络地址。

2.2 Qt

2.2.1 Qt介绍

UI(User Interface)也可以称之为用户接口或使用者接口，是系统和用户之间进行交互和信息交换的媒介，它实现信息的内部形式与人类可以接受形式之间的转换。

Qt[3]是由挪威 TrollTech 公司于 1995 年底出品，之后被诺基亚公司收购并维护的一个跨平台的 C++图形用户界面应用程序框架。广泛用于开发 GUI 程序，这种情况下又被称为部件工具箱。也可用于开发非 GUI 程序，比如控制台工具和服务器。自从 1996 年早些时候，Qt 进入商业领域，它已经成为全世界范围类数千种成功的应用程序的基础，被 Google Earth、KDE、西门子公司等使用。它是诺基亚（Nokia）的 Qt Development Frameworks 部门的产品。

经过多年发展，Qt 不但拥有了完善的 C++图形库，而且近年来的版本逐渐集成了数据库、OpenGL 库、多媒体库、网络、脚本库、XML 库、WebKit 库等等，其核心库也加入了进程间通信、多线程等模块，极大的丰富了 Qt 开发大规模复杂跨平台应用程序的能力，真正意义上实现了其研发宗旨“Code Less; Create More; Deploy Anywhere.”。

2.2.2 Qt 模块

模块化 Qt C++ 类库提供一套丰富的应用程序生成块 (block)，包含了构建高级跨平台应用程序所需的全部功能。Qt 模块结构图如图 2.1：

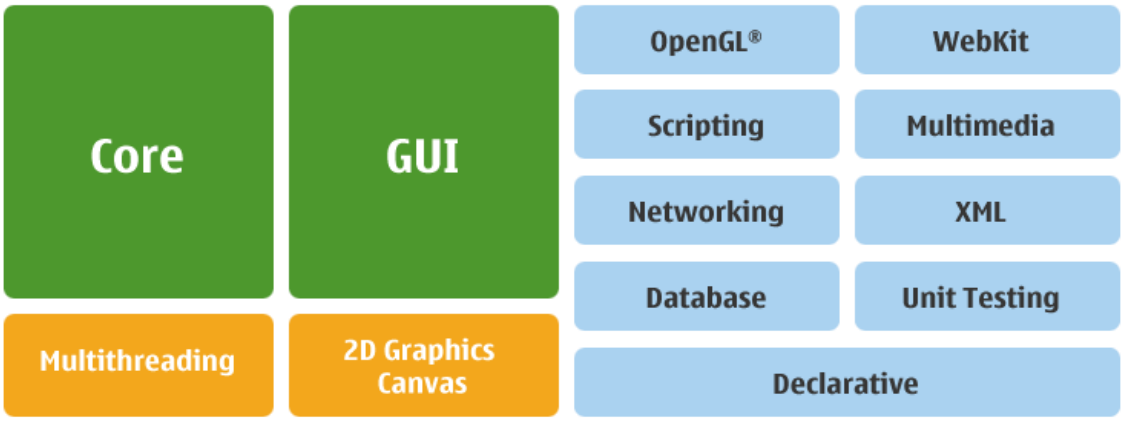

图 2.1 Qt 模块图

其中本系统设计到更多的 Qt 的消息响应机制和多线程如下：

（1）Qt 的消息响应机制：信号与槽：

Qt 利用信号与槽（signals/slots）机制取代传统的 callback 来进行对象之

间的沟通。当操作事件发生的时候，对象会发提交一个信号（signal）；而槽（slot）则是一个函数接受特定信号并且运行槽本身设置的动作。信号与槽之间，则通过QObject 的静态方法 connect 来连结。

信号在任何运行点上皆可发射，甚至可以在槽里再发射另一个信号，信号与槽的连结不限定为一对一的连结，一个信号可以连结到多个槽或多个信号连结到同一个槽，甚至信号也可连接到信号。

以往的 callback 缺乏类型安全，在调用处理函数时，无法确定是传递正确型态的参数。但信号和其接受的槽之间传递的数据型态必须要相符合，否则编译器会提出警告。信号和槽可接受任何数量、任何型态的参数，所以信号与槽机制是完全类型安全。

信号与槽机制也确保了低耦合性，发送信号的类并不知道是哪个槽会接受，也就是说一个信号可以调用所有可用的槽。此机制会确保当在"连接"信号和槽时，槽会接受信号的参数并且正确运行。

（2）多线程

Qt 的运行绪支持是独立于平台的运行绪类，采用信号与槽机制，实现类型安全的运行绪间通讯。这使得它易于开发具可移植性的多线程 Qt 应用程序。并能充分利用多核架构，获得最佳运行性能，还能根据可用的处理器内核数自动调整使用的运行绪数。多线程程序设计也是一个执行耗时操作而不会冻结用户界面的有效典范。

2.2.3 Qt的优势

（1）面向对象

Qt 的良好封装机制使得 Qt 的模块化程度非常高，可重用性较好，对于用户开发来说是非常方便的。Qt 提供了一种称为 signals/slots 的安全类型来替代 callback，这使得各个元件之间的协同工作变得十分简单。

（2）丰富的 API

Qt 包括多达 250 个以上的 C++类，还提供基于模板的 collections，serialization，file，I/Odevice，directorymanagement，date/time 类。甚至还包括正则表达式的处理功能。

（3）优秀的跨平台能力

使用 Qt 开发的软件，相同的代码可以在任何支持的平台上编译与运行，而不需要修改源代码。会自动依平台的不同，表现平台特有的图形界面风格。

Linux/X11：用于 X Window System（如 Solaris、AIX、HP-UX、Linux、BSD）。支持 KDevelop 和 Eclipse IDE 集成

Mac：用于 Apple Mac OS X。基于 Cocoa 框架。支持 Universal Binary。支

持以 Xcode 编辑、编译和测试。

Windows：用于 Microsoft Windows。支持 Visual Studio 集成

Embedded Linux：用于嵌入式 Linux。可以通过编译移除不常使用的组件与功能。通过自己的视窗系统 QWS，不需依赖 X Window System，直接写入 Linux 帧缓冲。可以减少存储器消耗。并提供虚拟帧缓冲 QVFb，方便在桌面系统上进行嵌入式测试。

Windows CE / Mobile ：用于 Windows CE

Symbian：用于 Symbian platform

Maemo：用于 Maemo

（4）大量的开发文档

对于 Qt 的各个模块，都有对应的帮助文档和编写好的例子供开发人员参考。

3 系统需求分析

服务端程序运行流程图如图 3.1:

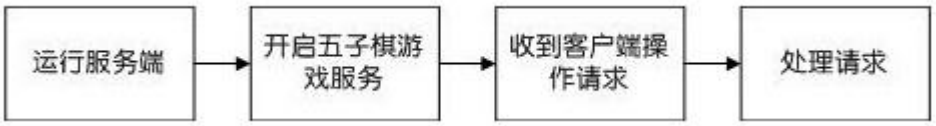

图 3.1 服务端需求分析流程图

打开服务端后，用户首先设置服务参数，然后等待客户端连接。

当收到客户端的连接时，将目前已经连接到服务端的所有客户端的状态（正在游戏中，建立了主机并等待玩家加入）发送到刚刚连接进来的客户端。

如果收到客户端请求：请求建立游戏主机、请求加入别人的游戏主机进行游戏，则服务端更新此客户端的状态，然后将最新的状态发送给各个客户端。

客户端程序运行流程图如图 3.2：

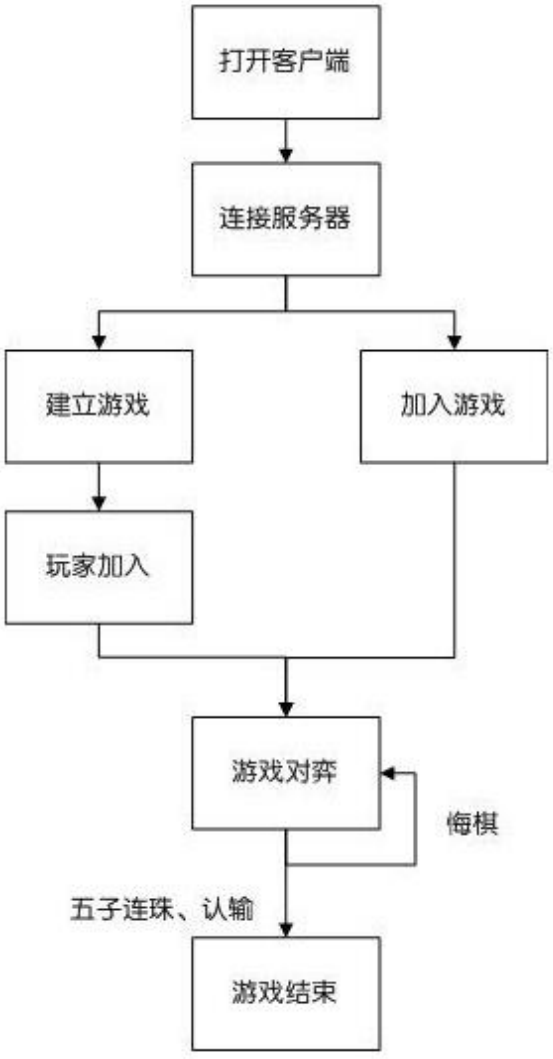

图 3.2 客户端需求分析流程图

打开客户端后，用户首先设置服务器参数，然后连接服务端。连接成功后，客户端会收到来自服务端的其他客户端的当前状态，用户可以选择是建立游戏主机，还是加入别人的游戏主机进行游戏。

（1） 功能需求

程序主要核心功能如图 3.3：

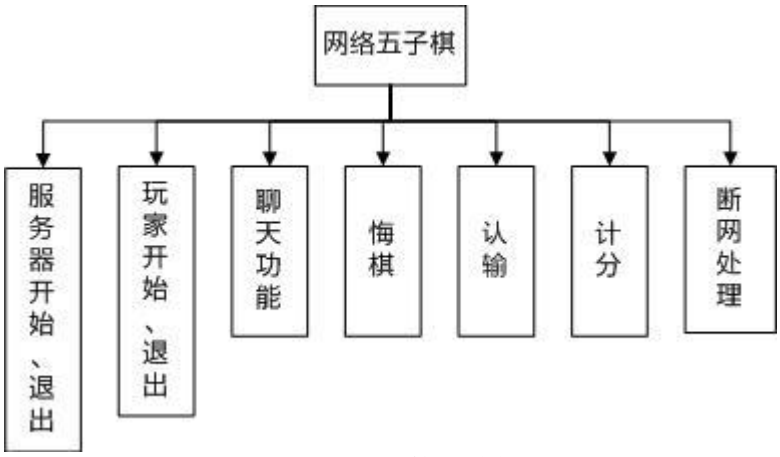

图 3.3 程序核心功能

- 服务器开始、退出

服务端程序能够开启、关闭五子棋游戏的服务。关闭服务时，能够给当前已连接的客户端发送关闭服务消息。

- 玩家开始、退出

玩家进入游戏主机之后，需要点击准备才能开始游戏，若玩家在游戏时退出游戏，需要向对手发送退出游戏的信息。

- 聊天功能

玩家在对弈过程中可以发送信息进行交流。

- 悔棋

玩家在对弈过程中，能够进行悔棋请求，如果对方答应悔棋请求，则系统撤销双方玩家最近一次的落子；若对方不答应悔棋请求，则游戏继续。

- 认输

玩家在对弈过程中，可以直接选择认输，不需要经过对方同意。

- 计分

程序能保存当前两个玩家从开始第一句对弈，到其中一方离开游戏这期间的战绩情况。

- 断网处理

当其中一方玩家意外断网时，程序能通过超时相应判断出对方丢失了网络连接并作出处理。

（2）环境需求

- 装有windows或者linux的操作系统的计算机两台
- 计算机的CPU不低于P4，内存不小于64M
- 10/10Mbps自适应网卡每台计算机各一块
- 每台计算机都应配置有TCP/IP协议

（3）用户界面需求

- 界面友好、美观
- 界面简洁
- 操作界面方便、易懂

（4）异常处理需求

- 处理端口号冲突时的异常
- 当出现错误落子位置时，此次操作无效
- 如果网络非正常断开，则终止此局游戏

4 系统设计

4.1 系统设计思想

本系统的能够实现一个在网络上供玩家进行对战的网络版的五子棋游戏，玩家只要登陆到服务器上，然后选择其它已创建好的游戏主机，进行五子棋对战游戏。或者自己建立一个游戏主机，等待对方的加入。

在C/S 模式游戏中，服务端一般提供所有用户的全局信息，并能提供客户之间的信息转发，客户之间的通讯必须通过服务端进行。因为在多个客户能够连接到同一台服务端上，所以服务端必须用线程负责每个用户的通讯和消息处理。

但是考虑到如果采用这种思路，当客户端达到一定的数量之后，会增加服务端的负荷，而且当两个玩家开始游戏之后，服务端只需知道玩家双方是否退出游戏以及双方的游戏结果即可，而玩家之间的对弈信息、聊天等信息大可不必经过服务端。

基于以上考虑的原因，决定采用P2P的网络模式：服务端程序开启服务后，通过一个线程监听客户端的连接，一旦有客户端连接，服务端便为该客户端建立连接并启动一个特定的线程，利用该连接不断获取客户端操作请求，从而更新游戏大厅信息，让其他玩家及时了解到哪些客户端建立了游戏主机，哪些客户端之间正在进行游戏对弈。而当客户端加入了另外一个客户端建立的游戏主机时，二者便建立起P2P连接，用于传送二者之间的对弈信息，操作请求（悔棋，认输等），聊天信息等。设计思想的结构拓扑图如图4.1：

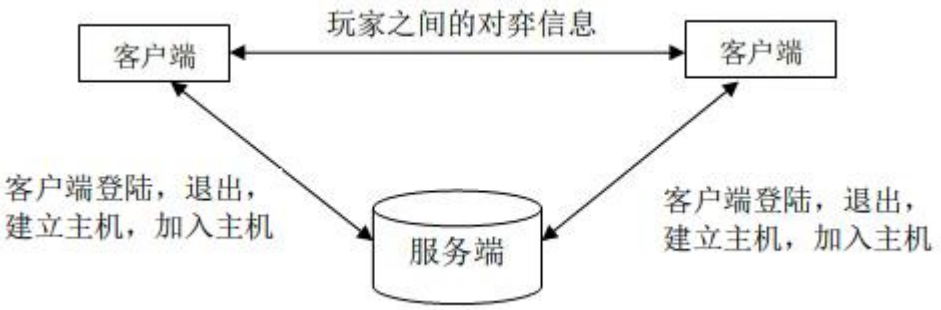

图4.1 系统结构拓扑

4.2 系统总体设计

系统使用 Socket 技术与 Qt 多线程机制结合，使用 C/S 模式与 P2P 模式结合，在进行客户与服务端以及各个客户端之间进行信息的交互。

其中，对客户端进行的各个操作请求产生的数据包发送的对象如表 4.1：

表 4.1

客户端操作类型	数据包发送对象
用户登录请求	服务器
用户建立主机请求	服务器
用户加入游戏请求	服务器
网络丢失确认	服务器
对方网络丢失信息通知	服务器
发送认输通知	客户端
发送落子信息	客户端
发送聊天内容	客户端
发送悔棋请求	客户端

用户程序之间进行交互的协议如图 4.2：

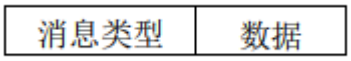

图 4.2 用户程序交互协议

其中，“数据类型”字段如表 4.2：

表 4.2

消息类型	说明
COMM_SERVER_CONN_FAILED	连接服务器失败
COMM_SERVER_CONN_SUCCESSFUL	连接服务器成功
COMM_SERVER_CLOSE	服务器关闭
COMM_SERVER_GAMEINFO	服务器发送大厅信息
COMM_CLIENT_QUITGAME	客户端退出游戏
COMM_CLIENT_DISCONN	客户端请求断开服务器
COMM_CLIENT_CONN	客户端请求连接服务器
COMM_CLIENT_CREATE	客户端请求创建主机
COMM_CLIENT_JOIN	客户端请求加入游戏
COMM_CLIENT_GAMESTART	双方准备完毕，游戏开始
COMM_CLIENT_GAMEOVER	某方玩家胜利，游戏结束
COMM_CLIENT_GAMEOP	玩家游戏操作：落子
COMM_CLIENT_CHAT	玩家游戏操作：发送聊天信息
COMM_CLIENT_UNDO	玩家游戏操作：悔棋
COMM_CLIENT_UNDO_YES	玩家游戏操作：悔棋回复 yes

COMM_CLIENT_UNDO_NO	玩家游戏操作：悔棋回复 no
COMM_CLIENT_LOSE	玩家游戏操作：认输
COMM_CLIENT_LOSTCONN	玩家发送消息：对手超时，判定为掉线

请求方需要发出请求时，便把消息类型设置为如上对应的消息类型（例如：服务器连接请求），并设置请求相关的数据（例如：需要连接的服务器的 IP 以及端口），最后发送此数据包。

数据字段根据第一个字段的不同设置不同的数据，有可能为包含五子棋落子位置的信息；有可能为空数据（例如：连接服务器成功则数据字段为空，连接服务器失败则数据字段为连接服务器失败的原因）。

接收方在接受到此数据包之后，首先判断出数据包的类型什么，然后利用接受到的数据（例如：服务器的 IP 以及端口）以及数据包类型（例如：服务器连接请求）来处理数据（例如：是否接受此客户端的连接），处理完毕之后，设置响应数据包类型为具体的响应类型（例如：连接成功），并且设置响应相关的数据（例如：连接成功则没有数据，连接失败则数据为连接失败的原因），最后发送此数据包。

例如某个客户端在连接服务器时，详细流程如图 4.2：

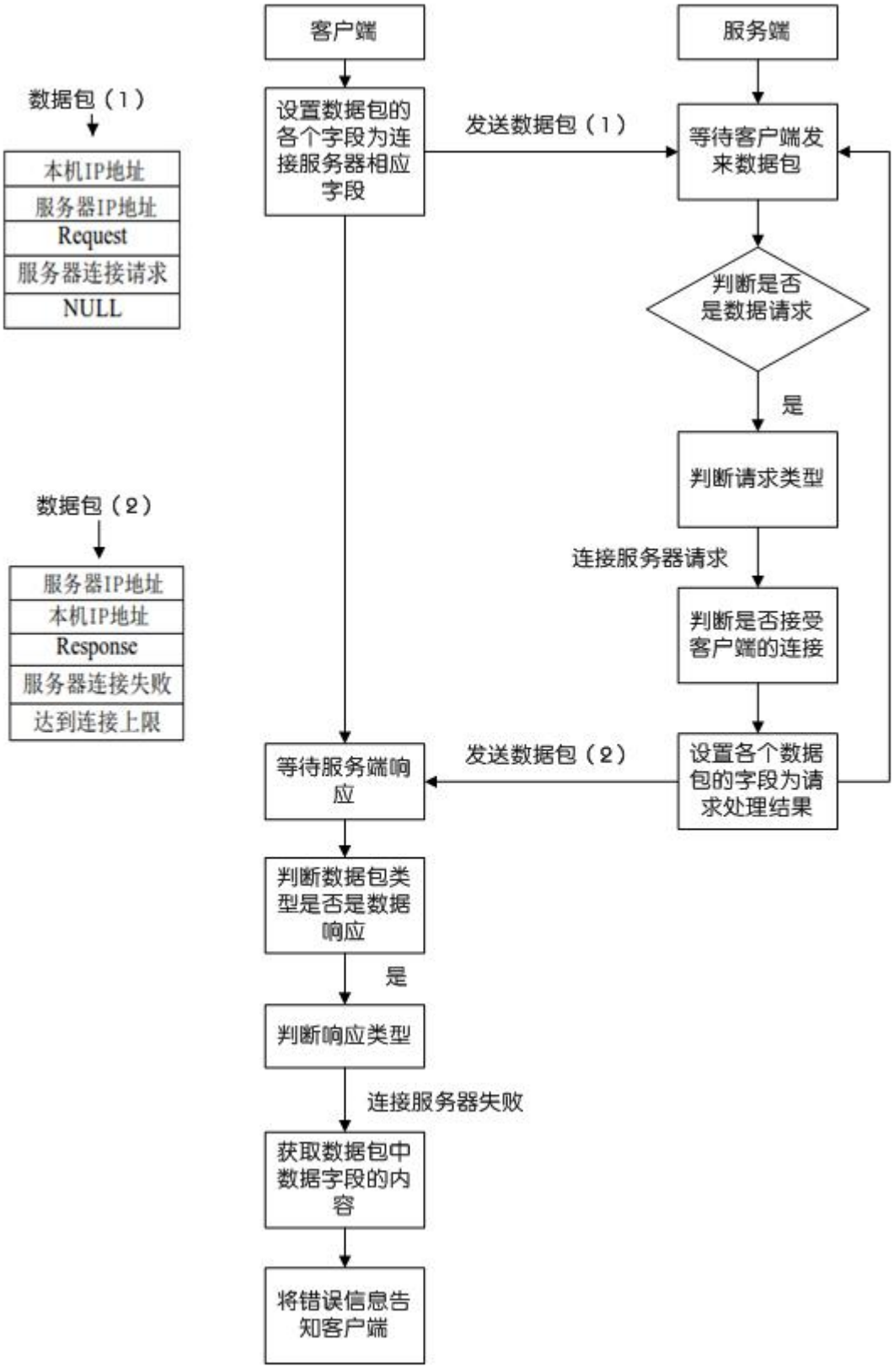

图 4.2 客户端连接服务端时的流程

客户端想服务端发送连接请求，数据包设置如图中“数据包（1）”，服务端收到连接请求之后，判断出此时已达服务端最大的连接数，然后将数据包设置如图

中“数据包（2）”，然后将此数据包发送给刚刚的客户端，通知客户端此次连接因为服务器已经达到了最大连接数，拒绝客户端的连接。

4.3 系统工作流程

4.3.1 服务端工作流程

基于以上设计，服务端只负责本地游戏服务的开启，以及各个客户端的连接，建立主机，加入游戏等请求，因此服务端的工作流程如图 4.3：

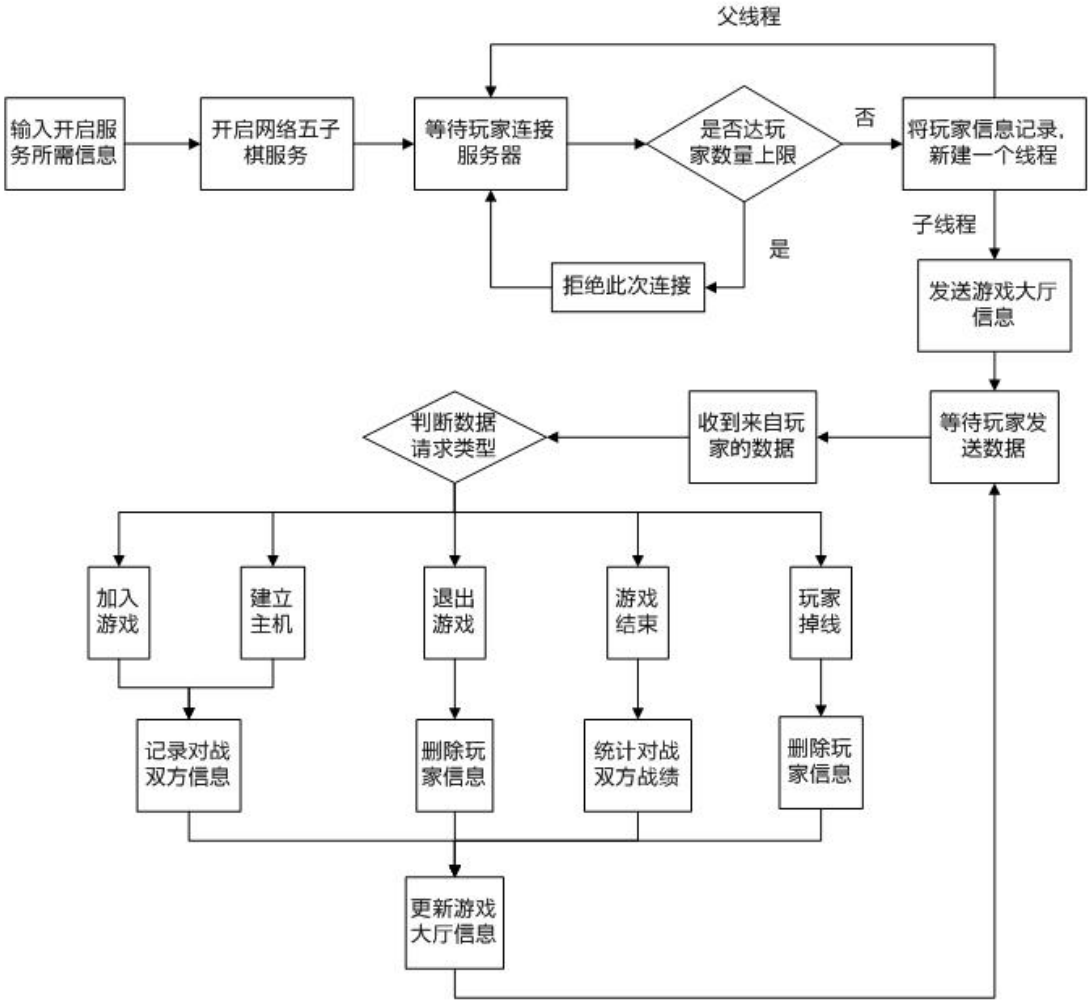

图 4.3 服务端工作流程图

服务端首先设置服务参数，然后开启本地服务，等待客户端的连接。

当接受到客户端的连接之后，建立一个新线程，用于接受客户端的操作请求，父线程则继续等待客户端的连接。

当子线程收到客户端发来的操作请求时，则根据对应的请求类型做相应的处理。

4.3.2 客户端工作流程

基于以上设计，客户在与服务端交互时采用 C/S 模式，与其他玩家对弈时，采用 P2P 模式，因此客户端的工作流程如图 4.4：

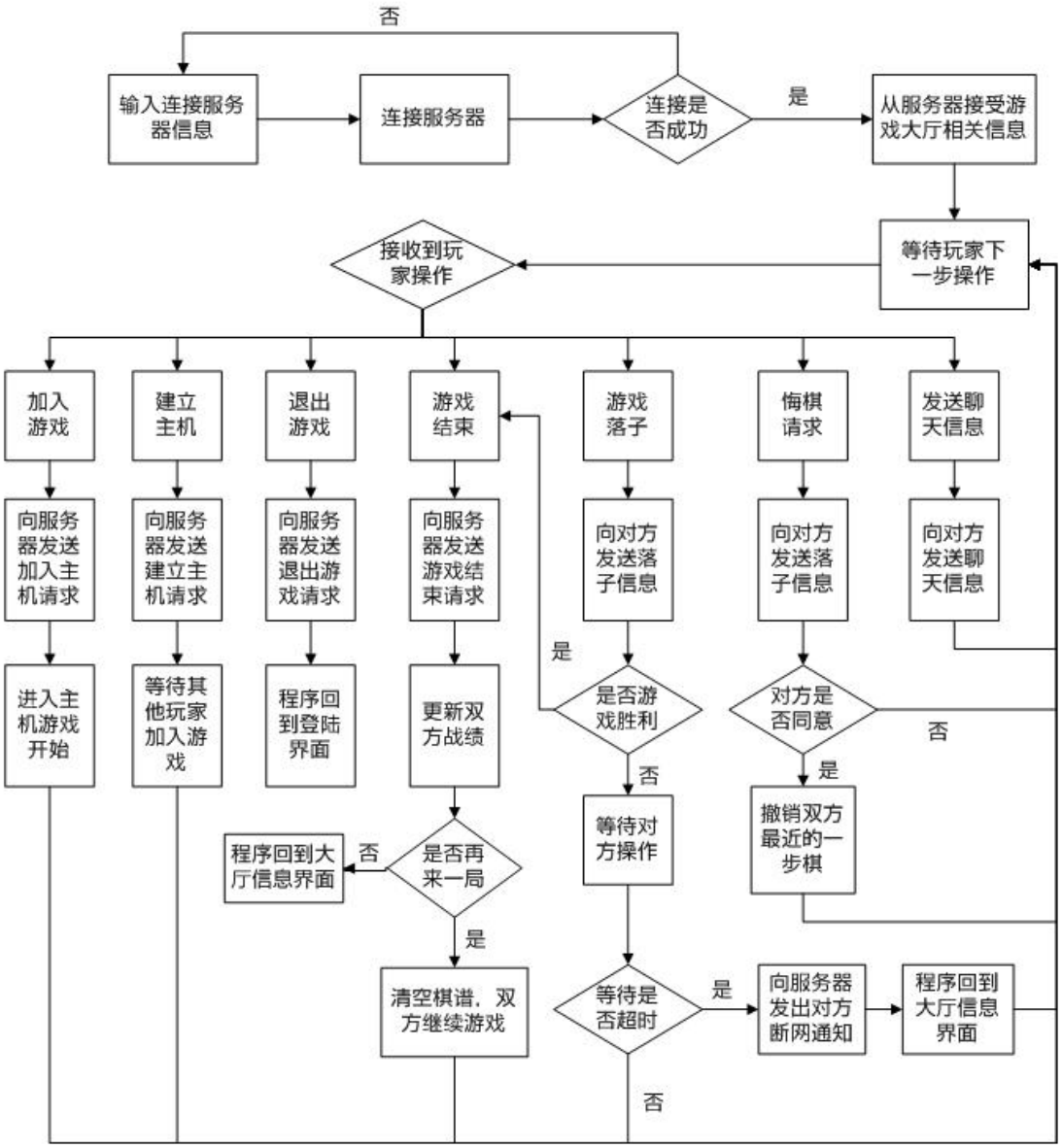

图 4.4 客户端工作流程图

客户端连接到服务端之后，首先获取到服务端发送来的各个客户端的状态，然后选择是自己建立游戏主机，还是加入别人的游戏。

游戏中，客户端与客户端之间会建立 P2P 连接，传递各种游戏操作信息。

4.4 五子连珠算法设计

五子棋游戏中，如果一方胜利，则胜利时的五子连珠必定与玩家最后落下的那颗棋子相关。因此每当用户落下一颗棋子时，程序便判断包括这颗棋子在内的纵向、横向、斜向的五颗棋子有没有五子相连的情况，若有，则此方胜利，反之，

继续比赛。

五子棋程序中，采用一个二维数组保存棋盘的每个位置的当前状况（白色棋子，黑色棋子，没有落子）。当其中一方胜利时，在棋盘记录中，五子相连的方向的当前状态应同为白子或同为黑子。

以斜向为例，胜利时的可能情况如图 4.5 所示：（所有圆均表示同一个玩家落下的棋子，填充色的圆表示最后落下的子）：

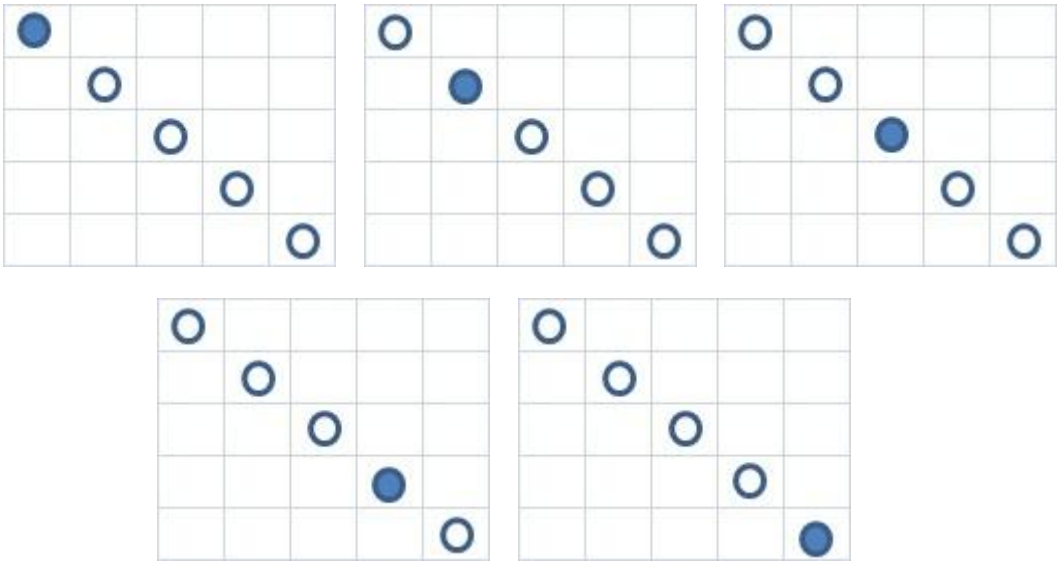

图 4.5 五子连珠算法

每次落下一颗棋子时，程序判断斜向这 5 种情况以及其他几个方向类似情况是否为同色，若是，则胜利，反之，继续比赛。

程序伪代码：

curPosX,curPosY ： 当前落子的横纵坐标

count ： 棋盘格数（19）

winColor ： 当前落子玩家所持有的棋子的颜色

chessboardRecord[count][count] ： 记录棋盘各个位置的落子的颜色的二维数组

```
for I from 4 to 1 step 1
    if curPosX<0 or curPosY<0 then
        continue;
    end if
    for J from 0 to 5 step 1
        if curPosX>count-1 or curPosY>count-1 then
            break
        end if
        if chessboardRecord[curPosX][curPosY]<>winColor
```

```
            break
        end if
        else if j=4
            return true
        end if
    end for
end for
```

5 系统实现

5.1 主要的数据结构

① enum playerRole

```
enum playerRole
{
    HOST,
    GUEST
};
```

用于记录某个棋子是主机落下的还是对手落下的。

② enum playerStatus

```
enum playerStatus
{
    playingGame,
    readyToPlayingGame
};
```

记录当前游戏状态是 playingGame（正在游戏）还是 readyToPlayingGame（等待对方按下准备按钮，然后开始游戏）。

③ enum comm_request_type

```
enum comm_request_type
{
    //连接服务器失败
    COMM_SERVER_CONN_FAILED,
    //连接服务器成功
    COMM_SERVER_CONN_SUCCESSFUL,
    //服务器关闭
    COMM_SERVER_CLOSE,
    //服务器发送大厅信息
    COMM_SERVER_GAMEINFO,

    //退出游戏
```

```
    COMM_CLIENT_QUITGAME,
    //请求断开服务器
    COMM_CLIENT_DISCONN,
    //客户端请求连接服务器
    COMM_CLIENT_CONN,
    //客户端请求创建主机
    COMM_CLIENT_CREATE,
    //客户端请求加入游戏
    COMM_CLIENT_JOIN,
    //双方准备完毕，游戏开始
    COMM_CLIENT_GAMESTART,
    //某方玩家胜利，游戏结束
    COMM_CLIENT_GAMEOVER,
    //玩家游戏操作：落子
    COMM_CLIENT_GAMEOP,
    //玩家游戏操作：发送聊天信息
    COMM_CLIENT_CHAT,
    //玩家游戏操作：悔棋
    COMM_CLIENT_UNDO,
    //玩家游戏操作：悔棋回复 yes
    COMM_CLIENT_UNDO_YES,
    //玩家游戏操作：悔棋回复 no
    COMM_CLIENT_UNDO_NO,
    //玩家游戏操作：认输
    COMM_CLIENT_LOSE,
    //玩家发来消息，对方掉线
    COMM_CLIENT_LOSTCONN
};
```

用户程序之间交互协议。

④ struct msg_request_struct

```
struct msg_request_struct
{
    //请求类型
```

```
    qint8 request;
    //数据
    QString data;
};
```

用户程序之间交互数据包的格式。

5.2 服务端的实现

5.2.1 界面实现

服务端能够开启本地服务，以及实时查看游戏大厅信息：哪些玩家正在对弈，哪些玩家在等待其他玩家加入等。因此服务端应具有两个界面，分别用来配置本地服务以及显示游戏大厅信息。如图 5.1 和图 5.2：

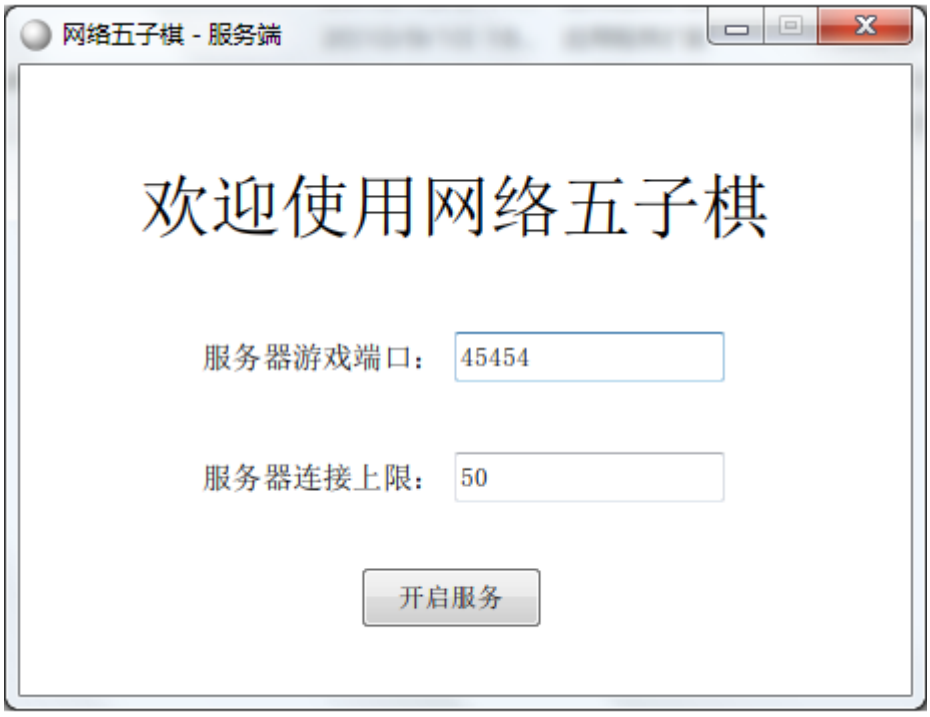

图 5.1 服务端配置界面

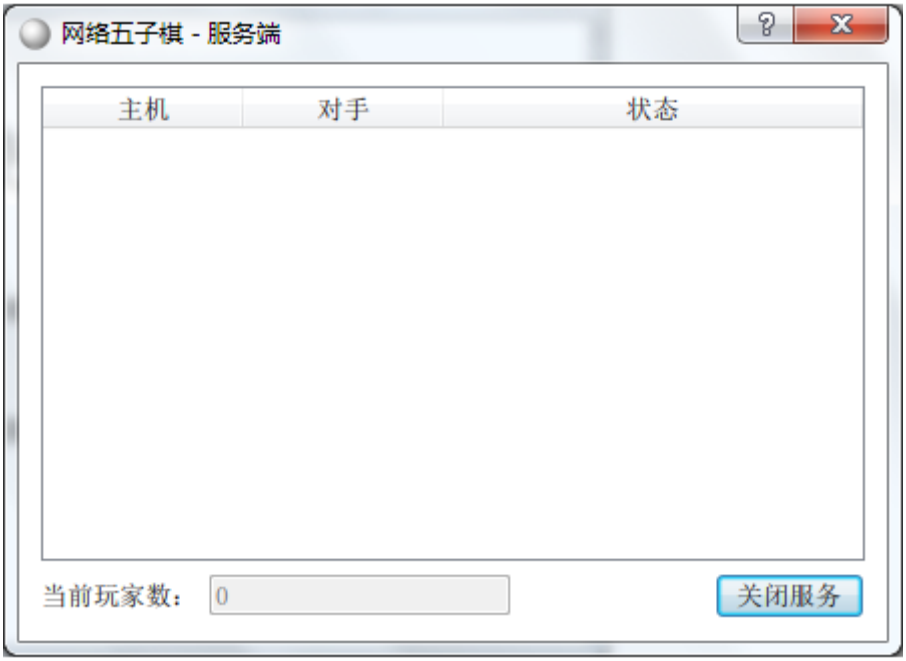

图 5.2 游戏大厅信息

5.2.2 主要的成员变量

① msg_request_struct* msg_req_struct;

用户程序之间交互的数据结构。

② QList<QPair<QString, QString> >playerFightInfo;

记录对弈双方的 IP 信息，若主机在等待其他玩家加入，则其对手 IP 地址置为“-”。

③ QTcpServer *tcpServer_player;

保存服务端本地开启的服务套接字。

5.2.3 功能实现

①客户端连接限制功能

考虑到服务端运行环境不同，所能承受的负荷也不同，因此加入“客户端连接限制”功能，防止过多的客户端连接导致服务端的负荷过大。

在服务端的服务配置界面，可以设定服务端允许的最大连接数，当客户端的连接数达到了服务端设置的最大连接数的时候，服务端便不允许其他客户端连接。

核心代码：

```
QTcpSocket *tcpSocket_player = tcpServer_player->nextPendingConnection();
DataClass::curConnCount++;
if(DataClass::curConnCount > DataClass::maxConnectionLimit)
{
    DataClass::curConnCount--;
```

```
        tcpSocket_player->close();
        tcpSocket_player = NULL;
        return ;
    }
```

②广播游戏大厅信息（建立主机，加入游戏，退出游戏等）

服务端与客户端之间的信息交互采用 C/S 模式，因此当有客户端更新并向服务端发送信息（例如建立主机，加入游戏，退出游戏等）之后，服务端应及时将更新之后的信息广播出去，使得其他客户端及时更新大厅信息。

核心代码：

```
QString qsNull = "-";
playerFightInfo.push_back(qMakePair(clientAddr, qsNull));
updateGameInfo();
QList<QTcpSocket* > allTcpSocket =
    tcpServer_player->findChildren<QTcpSocket *>();
QString data;
for(int i=0; i<playerFightInfo.size(); i++)
    data += playerFightInfo[i].first + " " + playerFightInfo[i].second + "_";
for(int i=0; i<allTcpSocket.size(); i++)
{
    DataClass::sendMsg(COMM_SERVER_GAMEINFO, data, tcpSocket);
}
```

客户端收到服务端发送过来的 data 之后，用正则表达式分析出对弈双方的信息，然后更新本地显示。

5.3 客户端的实现

5.3.1 界面实现

客户端具有一个配置连接服务器的界面，用于输入连接服务器信息，如图 5.3：

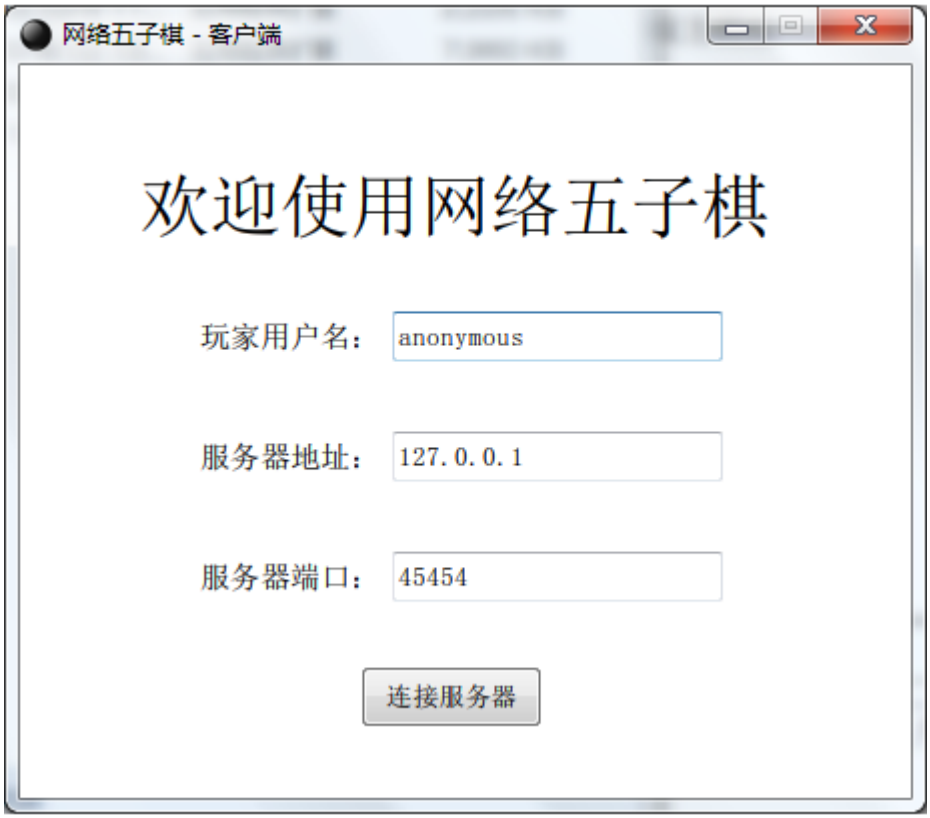

图 5.3 客户端连接服务端界面

登陆服务器成功之后，会显示从服务器接受到得玩家对弈信息，用户这时可以选择建立游戏主机等待别人加入，或者是加入别人的主机进行对弈，如图 5.4：

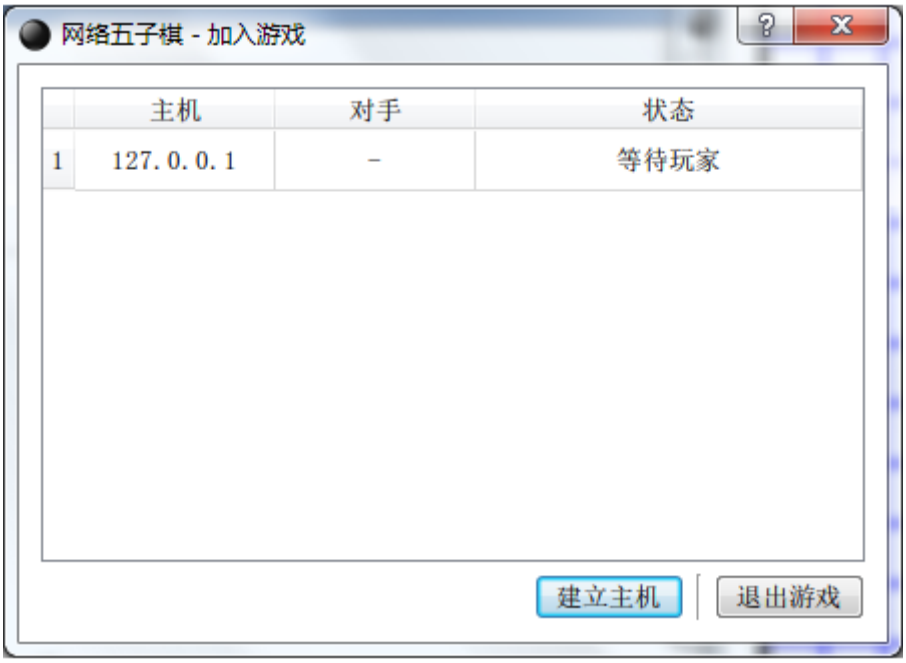

图 5.4 游戏大厅信息界面

玩家选择了建立游戏主机或者加入游戏后，会弹出游戏界面等待开始游

戏，如图 5.5：

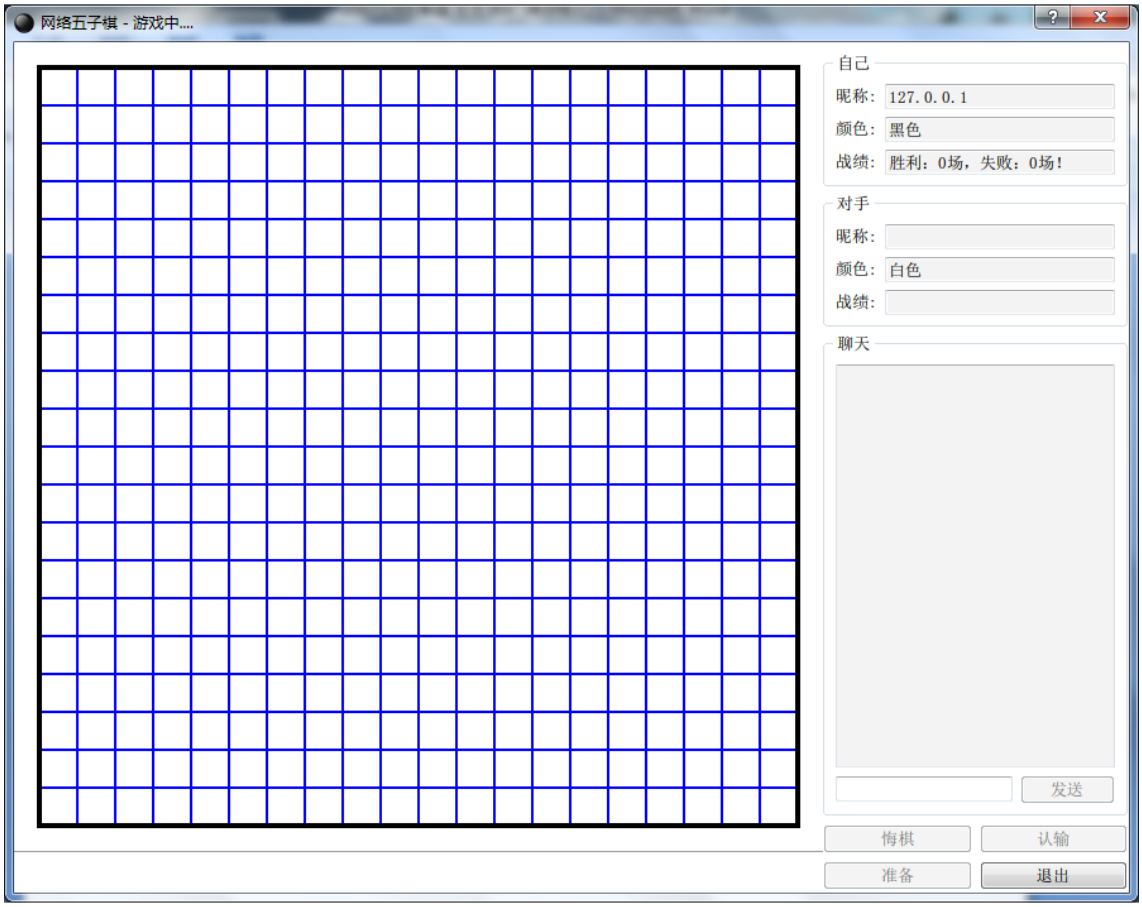

图 5.5 游戏主界面

5.3.2 主要的成员变量

① QTimer timer

记录超时时间的定时器。

② int chessboardRecord[19][19]

记录当前棋盘的落子信息。

③ msg_request_struct* msg_req_struct

用户程序之间交互的数据结构。

④ playerRole plyRole

当前玩家的身份（参考 5.1 的 playerRole 数据结构）。

⑤ playerStatus plyStatus

当前游戏状态（参考 5.1 的 playerStatus 数据结构）。

⑥ playerRole whosTurn

下一步该谁落子（参考 5.1 的 playerRole 数据结构）。

⑦ int win, lose

记录对战成绩。

⑧ QList<QPair<QPoint, enum playerRole> > pieceRecord

记录当前已经落下的棋子的信息：落子的位置，谁落的这颗子。

5.3.3 功能实现

① 玩家开始、退出

开始时输入玩家的姓名，同时下方会显示出相应的 IP 地址。在输入相关的信息后就会弹出游戏界面，而游戏设置的对话框随即会自动隐藏，释放。这时就开始等待对方的加入，等有人加入后就可以开始游戏了。或者直接加入别人建立的主机。如果不想玩了就可以按退出按钮，一方退出游戏后，另一方就会显示尚未连线，然后等待下一个玩家的进入。功能详细流程如图 5.6：

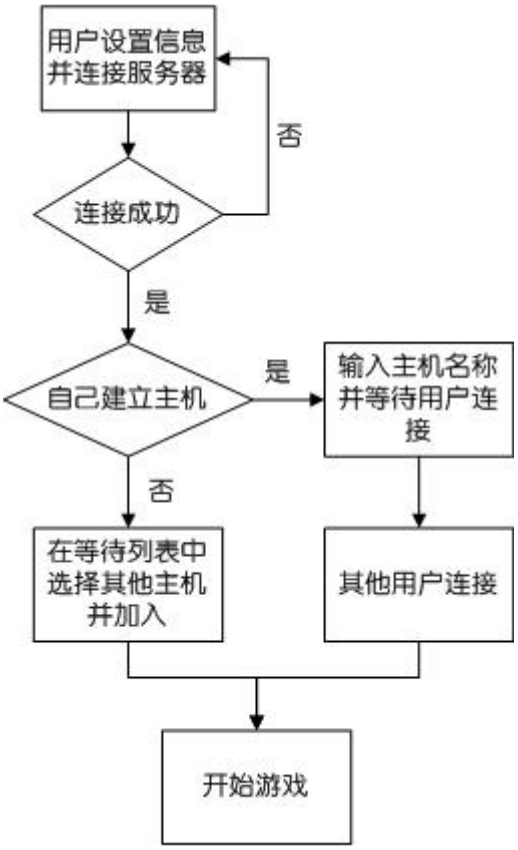

图 5.6 玩家开始、退出游戏功能

核心代码：

建立游戏主机：

```
DataClass::sendMsg(COMM_CLIENT_CREATE, "", tcpSocket_server);
GameStatus gameStatus;
gameStatus.setPara(tcpSocket_server);
connect(tcpSocket_server, SIGNAL(readyRead()),
        &gameStatus, SLOT(getNewDataFromServer()));
gameStatus.setPlayerRole(HOST);
```

```
gameStatus.exec();
```

加入游戏：

```
int row = ui.clientStatus_client_TBW->currentRow();
if(ui.clientStatus_client_TBW->item(row,1)->text() != "-")
{
    QMessageBox::information(this,"错误","此主机正在对战中...");
    return ;
}
DataClass::sendMsg(COMM_CLIENT_JOIN,
        ui.clientStatus_client_TBW->item(row, 0)->text(), tcpSocket_server);
GameStatus gameStatus;
gameStatus.setPara(tcpSocket_server);
connect(tcpSocket_server, SIGNAL(readyRead()),
        &gameStatus, SLOT(getNewDataFromServer()));
gameStatus.setPlayerInfo(ui.clientStatus_client_TBW->item(row,0)->text());
gameStatus.setPlayerRole(GUEST);
gameStatus.exec();
```

退出游戏：

```
if(plyRole == HOST)
{
    plyStatus = readyToPlayingGame;
    ui.chatLog_client_TE->clear();
    ui.record_me_client_LE->setText("胜利：" +
        QString::number(win) + "场，失败：" + QString::number(lose) + "场！");
    ui.record_rivar_client_LE->setText("");
    ui.addr_rivar_client_LE->setText("");
    QMessageBox::information(NULL,"提示","对方退出游戏！");
}
else if(plyRole == GUEST)
{
    ui.gameReady_client_BTN->setDisabled(false);
    closeByServer = true;
```

```
    this->close();
}
```

② 用户之间聊天

在下棋的同时可以通过聊天窗口和对方进行聊天会话，写完信息之后点击发送，就可以在公共的显示栏中看到自己发出的信息和对方回应的信息。进而开始用户之间的交流，同时所用的组合框的下拉菜单中会有一些简便的备用的语句，可以点击随即发送，这样就能省去部分打字的时间。

核心代码：

```
if(ui.sendwords_client_LE->text() == "")
{
    QMessageBox::information(NULL,"错误","请先输入文字！");
    return ;
}
ui.chatLog_client_TE->append("自己：   " + ui.sendwords_client_LE->text());
DataClass::sendMsg(COMM_CLIENT_CHAT,
        ui.sendwords_client_LE->text(),tcpSocket_player);
ui.sendwords_client_LE->setText("");
```

③悔棋、认输

在游戏中，玩家可以发出悔棋请求，如果对方同意悔棋，则程序自动回退一步；若对方不同意悔棋，则请求无效。玩家也可以发出认输请求，程序自动结束本轮游戏并计分，认输请求不需要对方同意。悔棋功能详细流程如图 5.7：

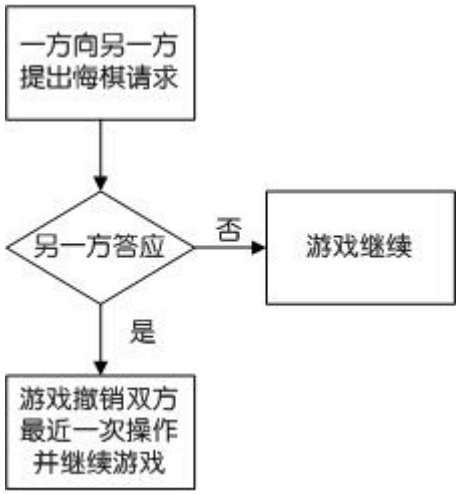

图 5.7 悔棋、认输功能

核心代码：

悔棋：

```
if(whosTurn != plyRole)
{
    QMessageBox::information(NULL,"错误",
                "轮到你下棋的时候你才能发出悔棋请求！");
    return ;
}
if(!(DataClass::checkDialog("确认","确认悔棋？")))
    return ;
whosTurn = (plyRole == HOST ? GUEST : HOST);
DataClass::sendMsg(COMM_CLIENT_UNDO,"",tcpSocket_player);
```

收到悔棋响应（同意悔棋）：

```
QMessageBox::information(NULL,"提示","对方同意悔棋！");
whosTurn = plyRole;
if(pieceRecord.size() > 0)
{
    chessboardRecord[pieceRecord[pieceRecord.size() - 1].first.x()-1]
        [pieceRecord[pieceRecord.size() - 1].first.y()-1] = 0;
    pieceRecord.removeAt(pieceRecord.size() - 1);
}
if(pieceRecord.size() > 0)
{
    chessboardRecord[pieceRecord[pieceRecord.size() - 1].first.x()-1]
        [pieceRecord[pieceRecord.size() - 1].first.y()-1] = 0;
    pieceRecord.removeAt(pieceRecord.size() - 1);
}
update();
```

收到悔棋响应（不同意悔棋）：

```
whosTurn = plyRole;
QMessageBox::information(NULL,"错误","对方不同意悔棋请求!");
```

认输：

```
if(!(DataClass::checkDialog("确认","确认认输？")))
    return ;
DataClass::sendMsg(COMM_CLIENT_LOSE,"",tcpSocket_player);
```

```
if(plyRole == GUEST)
    ui.gameReady_client_BTN->setDisabled(false);
ui.undo_client_BTN->setDisabled(true);
ui.lose_client_BTN->setDisabled(true);
plyStatus = readyToPlayingGame;
lose += 1;
ui.record_me_client_LE->setText("胜利：" +
    QString::number(win) + "场，失败：" + QString::number(lose) + "场！");
ui.record_rivar_client_LE->setText("胜利：" +
    QString::number(lose) + "场，失败：" + QString::number(win) + "场！");
```

④ 计分

游戏开始的时候双方的战绩均为零，当一轮过后战绩就会被刷新一次，各自几胜几败很清晰的出现在界面上，由于最先下棋的玩家胜利的几率会很大，所以根据各自的胜负情况会交替双方先手的顺序，达到公平的目的。

⑤ 断网处理

当客户端在等待对方操作时超时，则测试是否能连接服务器。若能连接，则向服务器发送对方以断网的相关信息并结束比赛计算分数；若不能连接，则提示用户已丢失与服务器和对方的连接。断网处理的详细流程如图 5.8：

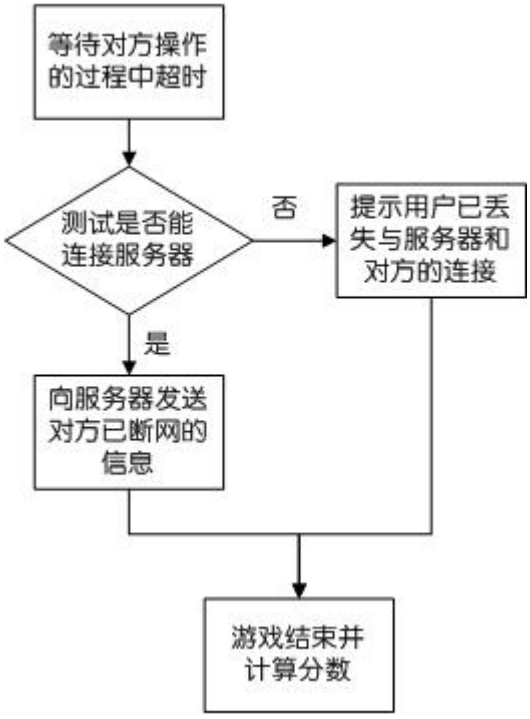

图 5.8 断网处理功能

核心代码：

```
QTimer timer;
connect(&timer, SIGNAL(timeout()), this, SLOT(timeout()));
timer.start(30000);
if(tcpSocket_player->isValid())
   return ;

QMessageBox::information(NULL,"提示","对方超时!");
dataInit();
ui.undo_client_BTN->setDisabled(true);
ui.lose_client_BTN->setDisabled(true);
ui.send_client_BTN->setDisabled(true);
if(plyRole == GUEST)
{
   DataClass::sendMsg(COMM_CLIENT_LOSTCONN,"HOST " +
       ui.addr_rivar_client_LE->text(),tcpSocket_server);
   if(tcpSocket_player)
   {
       tcpSocket_player->close();
       tcpSocket_player = NULL;
   }
   this->close();
}
else if(plyRole == HOST)
{
   DataClass::sendMsg(COMM_CLIENT_LOSTCONN,"GUEST " +
       ui.addr_rivar_client_LE->text(),tcpSocket_server);
   plyStatus = readyToPlayingGame;
   ui.chatLog_client_TE->clear();
   ui.record_me_client_LE->setText("胜利：" + QString::number(win) +
       "场，失败：" + QString::number(lose) + "场！");
   ui.record_rivar_client_LE->setText("");
   ui.addr_rivar_client_LE->setText("");
   if(tcpSocket_player)
```

```
    {
        tcpSocket_player->close();
        tcpSocket_player = NULL;
    }
    if(tcpServer_player)
    {
        tcpServer_player->close();
        tcpServer_player = NULL;
    }
}
```

5.3.4 五子连珠算法

核心代码（4 个方向中，以斜向为例）：

checkWinLURD(int last_x, int last_y, int winColor)函数：

```
for(int j=4; j>-1; j--)
{
    if(last_x - j <0 || last_y - j < 0)
        continue;
    for(int z=0; z<5; z++)
    {
        if(last_x - j + z > 18 || last_y - j + z > 18)
            break;
        if(chessboardRecord[last_x - j + z][last_y - j + z] != winColor)
            break;
        else if(z == 4)
            return true;
    }
}
return false;
```

6 系统效果展示

6.1 服务端效果展示

服务端配置页面（默认为监听端口为 45454 端口，默认连接上限为 50），如图 6.1：

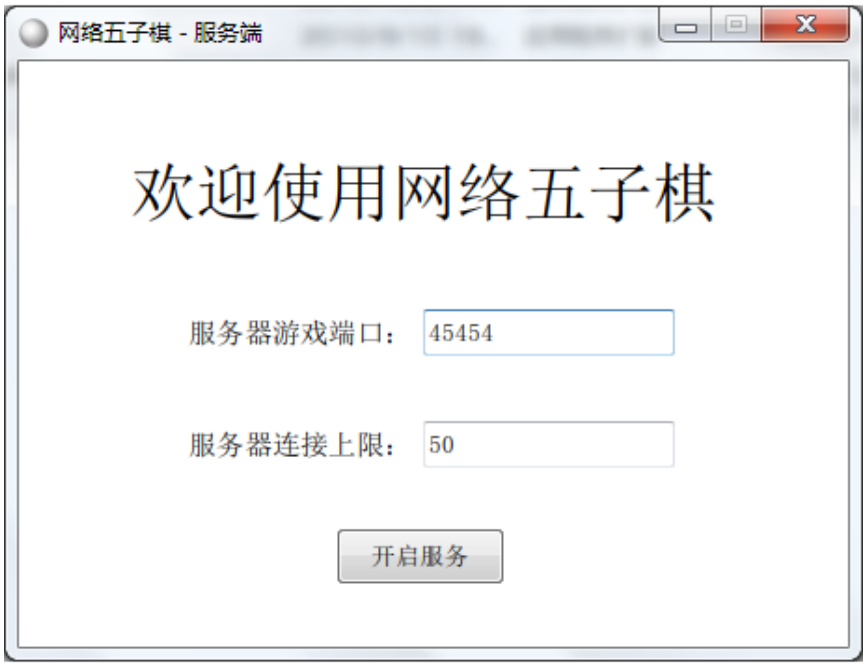

图 6.1 服务端配置界面

开启服务有，当有客户端连接时，左下角会实时更新当前连接数。当客户端建立游戏主机之，服务端也会及时显示，如图 6.2：

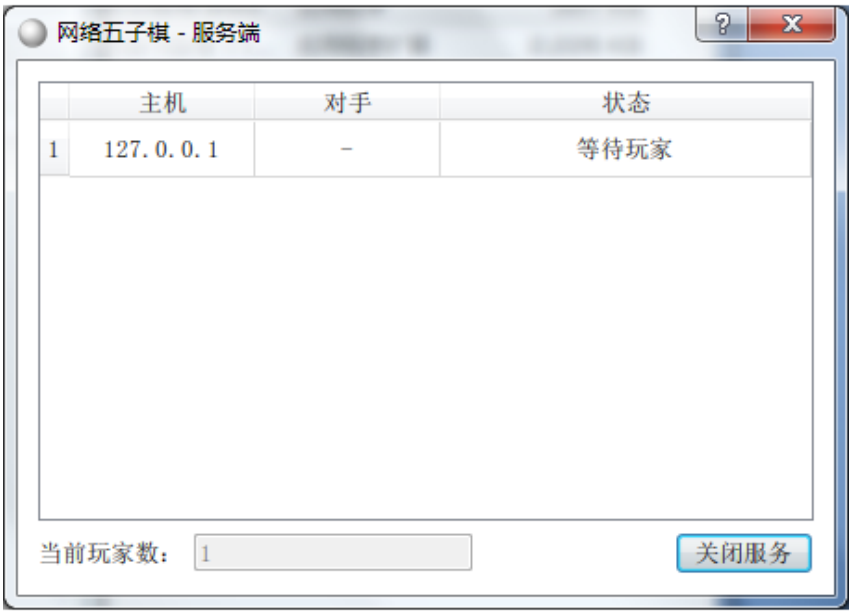

图 6.2 游戏大厅信息

当有玩家加入游戏时，服务端也能及时更新，如图 6.3：

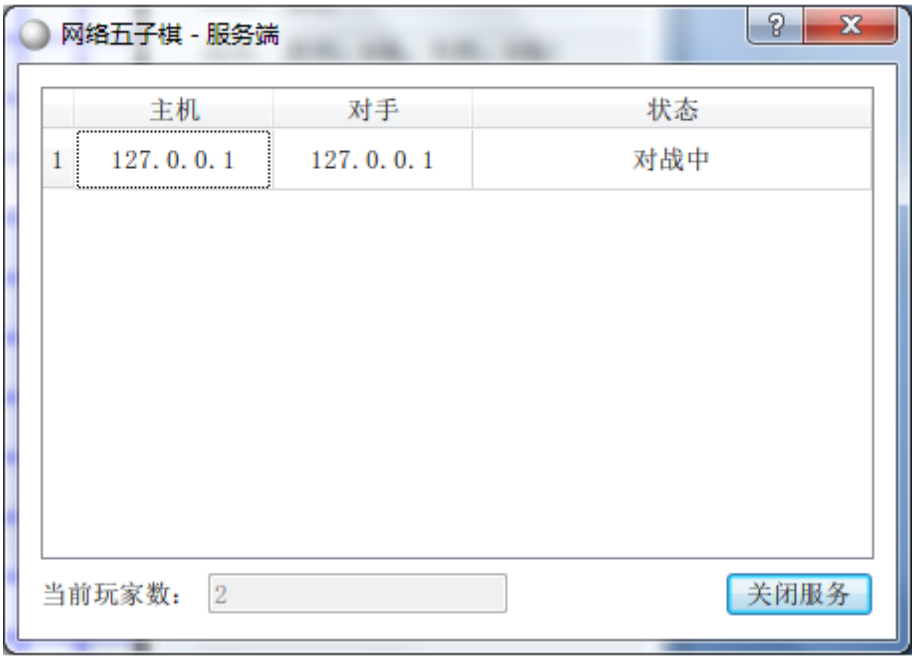

图 6.3 玩家加入游戏

6.2 客户端效果展示

客户端连接服务器配置页面，如图 6.4：

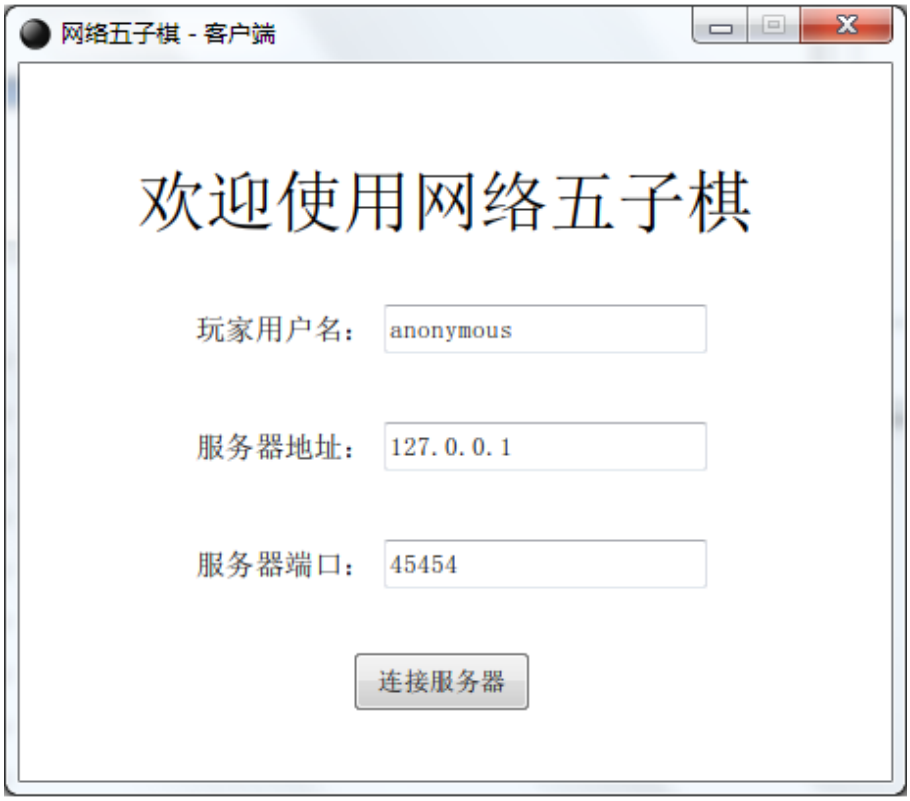

图 6.4 客户端配置服务器页面

客户端登陆后，会实时显示当前连接到服务端的游戏主机信息，如图 6.5：

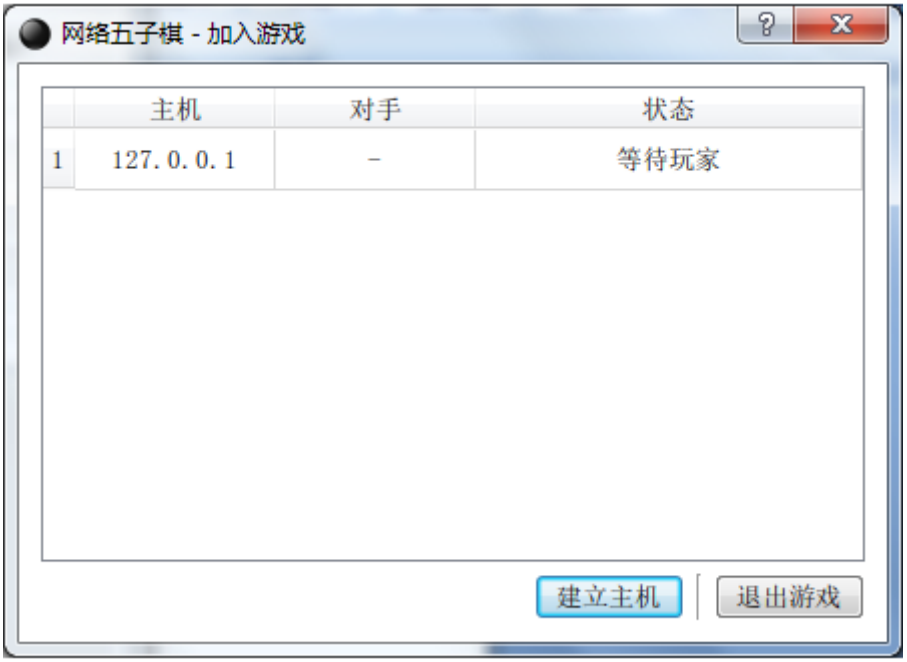

图 6.5 客户端实时显示当前游戏主机信息

玩家此时可以选择加入别人的游戏主机（双击主机列表），游戏界面如图 6.6：

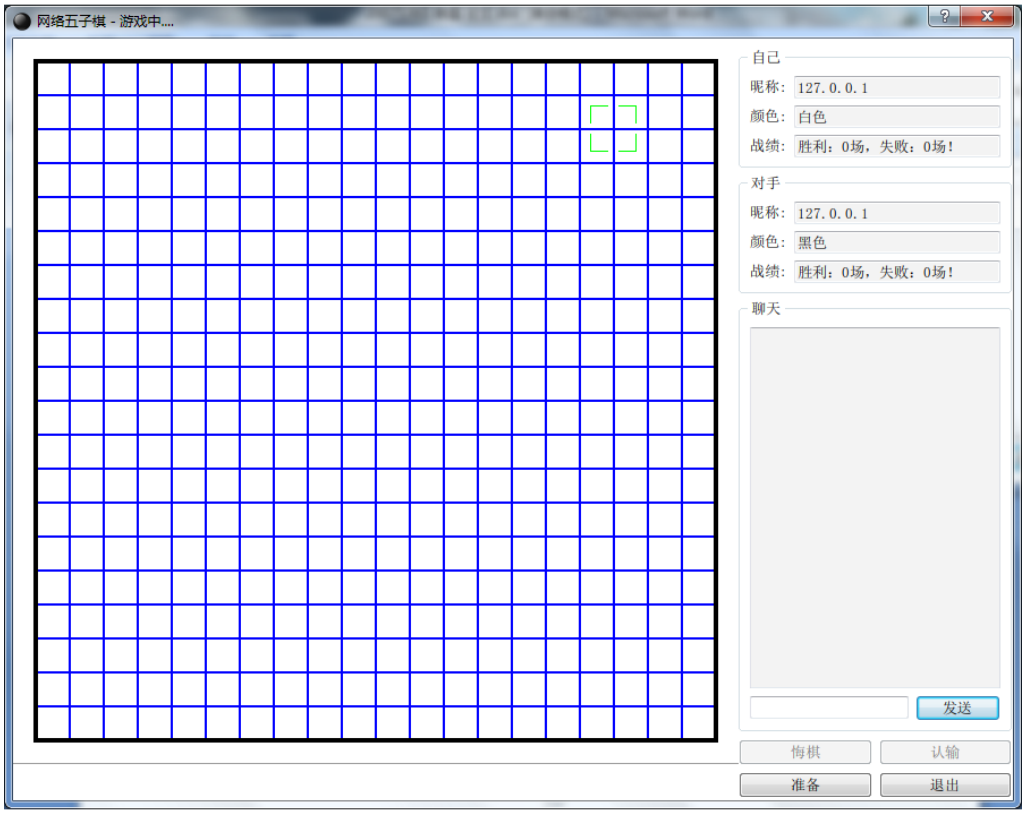

图 6.6 加入别人的游戏主机

点击“准备”按钮之后，即可开始游戏，如图 6.7：

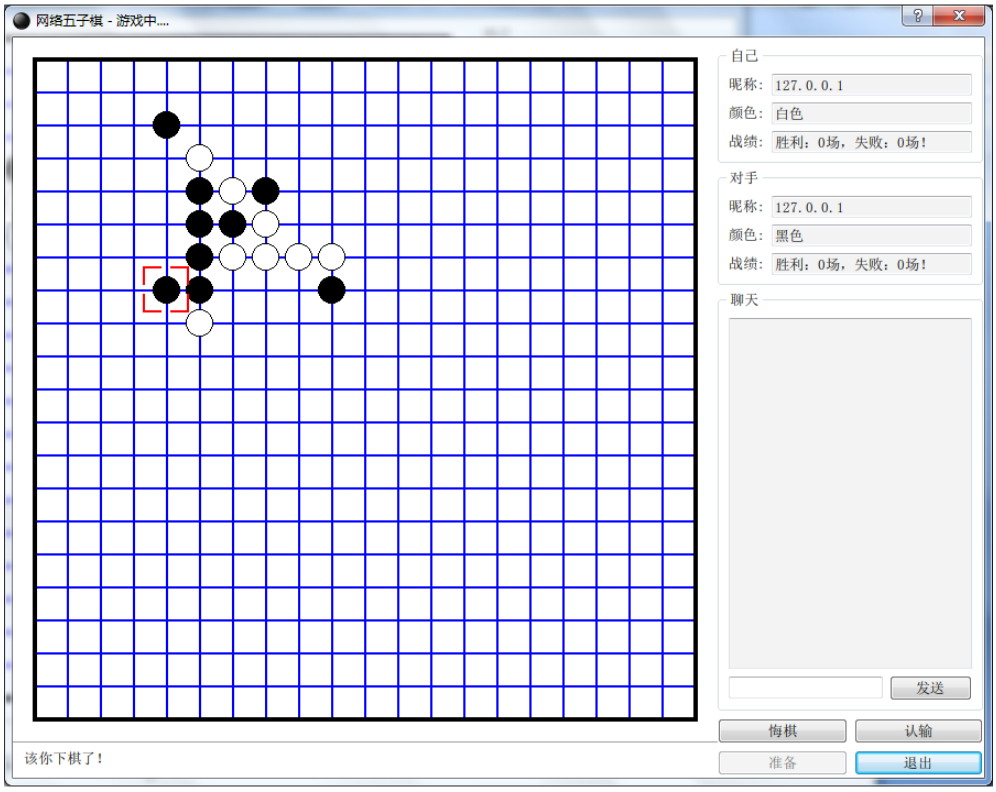

图 6.7 游戏界面

游戏过程中，玩家可以聊天，如图 6.8：

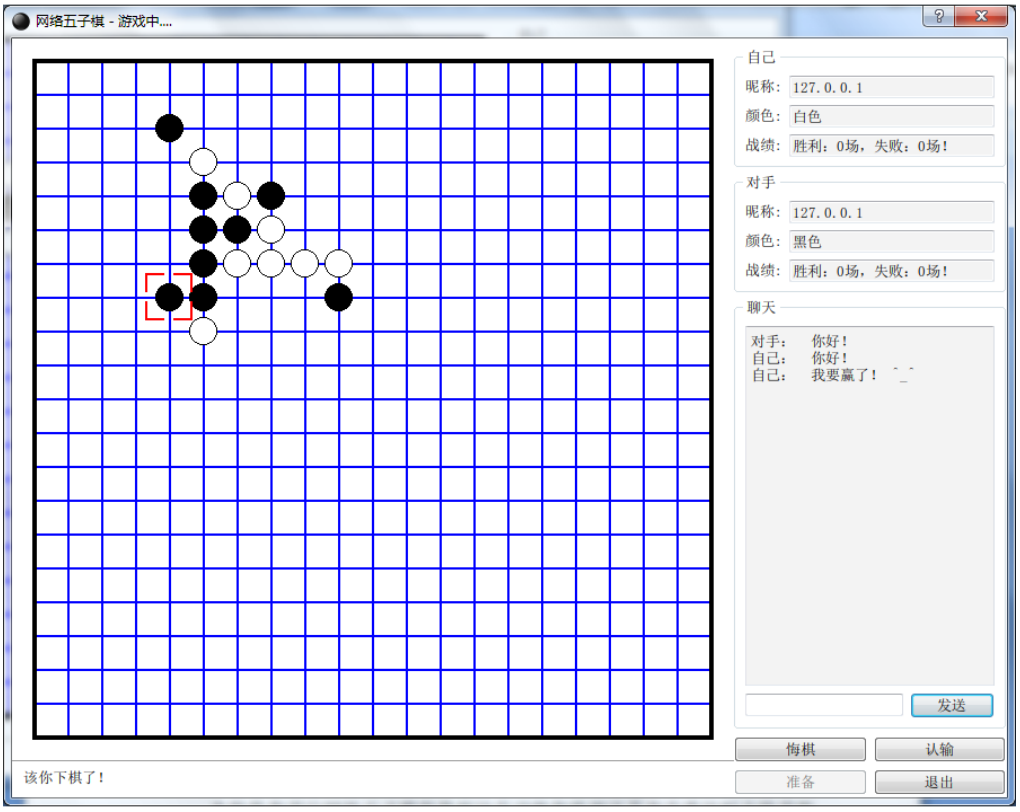

图 6.8 玩家聊天功能

游戏过程中，玩家可以点击“悔棋”、“认输”、“退出”按钮达到相应功能。

7 结论

7.1 已完成的工作

目前本软件已完成的工作如下：

（1）了解五子棋游戏的落子规则，并设计相应的判断胜利算法。

通过查阅相关资料，对于本软件设计了一个比较简单且快速有效的五子连珠算法。

（2）学习网络编程技术，掌握 C/S 模式网络程序的基本编程方法。

经分析，本软件最终采用了 P2P 的网络模式，相比于 C/S 模式，P2P 模式使得本软件效率更高。

（3）学习 GUI 程序设计，设计五子棋游戏的程序界面。

本软件已实现一个操作方便，界面美观的五子棋游戏程序。

（4）设计并实现网络五子棋的服务器和客户端程序。

本软件已实现服务器和客户端程序，服务端用于收集客户端的游戏状态信息，客户端用于玩家之间的对弈。

（5）设计并实现悔棋、认输、计分、聊天等游戏辅助功能。

本软件已实现常见的游戏功能：悔棋，认输，计分，聊天等功能。

另外，程序还实现了断网处理：当对方玩家 1 分钟之内没用做任何操作，则系统认为此玩家已丢失网络连接，结束此局比赛，并且让丢失连接的玩家退出此游戏主机。

7.2 下一步待完成的工作

本系统和其它的网络五子棋软件相比，功能上还存在着一定的差距，例如：本系统必须要知道主机的 IP 地址以及端口号才能建立连接、第三方观看的玩家不能加入棋局等等。

要想设计出更完善的网络五子棋软件，仍然需要做大量的工作：

①服务器的设置。

在网络中将其中一台计算机设置成为固定的服务器，其 IP 地址和端口号是固定不变的，客户端可以直接连接到此服务器上，然后通过服务器上的信息选择对手或者创建游戏主机，这样就省去了玩家必须知道游戏主机的 IP 地址和端口号的麻烦了。

②界面需要进一步美化。

界面可以适当进行美化，提高游戏的娱乐性。

③提高通信的可靠性。

在采用同 TCP 连接的一般情况下，网络五子棋软件都可以正常稳定的工作，但是仍不能保证其任何情况下都能正常稳定，一旦由于某种原因使得网络传输过程中的数据丢失，将使双方的游戏数据不一致，游戏便无法继续进行。因此，可以考虑一方断网之后，计算机智能代替下棋的功能。

④扩充功能。

增加记录当前棋谱的功能，以供五子棋残局爱好者能在日后研究。

致谢

从论文选题到搜集资料，从程序的设计到反复调试，从论文的成型到反复修改，期间经历了太多的喜悦、痛苦和彷徨。如今，伴随着这篇毕业论文的最终成稿，自己甚至还有一点成就感。

我要感谢我的导师刘骥老师。他为人随和热情，治学严谨细心。在闲聊中她总是能像知心朋友一样鼓励你，在论文的写作和措辞等方面她也总会以“专业标准”严格要求你，从选题、定题开始，一直到最后论文的反复修改、润色，刘老师始终认真负责地给予我深刻而细致地指导，帮助我开拓研究思路，精心点拨、热忱鼓励。正是刘老师的无私帮助与热忱鼓励，我的毕业论文才能够得以顺利完成，谢谢刘老师。

我要感谢计科三班的同学们。在百忙之中抽出时间帮助我搜集文献资料，帮助我理清论文写作思路，对我的论文提出了诸多宝贵的意见和建议。

参 考 文 献

[1] Wikipedia.五子棋[EB/OL]. [2011-04-07]. http://zh.wikipedia.org/zh/五子棋

[2] Wikipedia.点对点技术[EB/OL]. [2011-04-02].http://zh.wikipedia.org/wiki/点对点技术

[3] Wikipedia.Qt[EB/OL].[2011-04-03].http://zh.wikipedia.org/wiki/Qt

[4] Bruce Eckel.Thinking in Java[M].北京：机械工业出版社, 2007.

[5] Thomas H.Cormen，Charles E.Leiserson，Ronald L.Rivest，Clifford Stein.Introduction to Algorithms[M].北京：机械工业出版社, 2006.

[6] Jasmin Blanchette，Mark Summerfield.C++ GUI Qt 4[M].北京：机械工业出版社, 2009.

[7] 蔡志明.精通 Qt4 编程（第二版）[M].北京：机械工业出版社, 2011.

[8] 严蔚敏，吴伟民.数据结构（C 语言版）[M].北京：清华大学出版社, 2007.

[9] 张银犬. 基于 P2P 技术的信息资源共享模式研究[EB/OL].[2005-11-25]. http://epub.cnki.net/grid2008/detail.aspx?filename=TSGJ200505012&dbname=CJFD2005

[10] 顾军，王恒莉，徐丽. P2P 网络模型的分析与探讨[EB/OL].[2005-10-01]. http://epub.cnki.net/grid2008/detail.aspx?filename=JISJ200510015&dbname=CJFD2005

[11] 马睿. 基于 Qt 的 TCP 网络编程研究与应用[EB/OL].[2011-01-12]. http://epub.cnki.net/grid2008/detail.aspx?filename=FJDN201011067&dbname=CJFD2010