dotFive 设计文档

刘家昌
Jason Lau
计算机科学与技术系 42 班
Tsinghua CST 42
2014011307

M +86 185-1162-1217
E i@dotkrnl.com

简要描述

设计了一套用于通信的应用层协议，实现了一个可以通过网络进行对战的五子棋游戏。其中包括一个可以在各个平台下以服务形式稳定运行的服务器程序；以及与之对应的 GUI 客户端程序。

基本功能

实现了服务器与客户端之间的数据同步：根据要求，使用了 TCP 协议完成，并只使用了 QTcpServer 和 QTcpSocket 两个通信类。棋盘、请求等信息交互正常。

双方都按下开始键之后可以开始走棋：并且轮流落子功能、二十秒倒计时及超时弃权正常。

实现了悔棋、投降请求，在对方同意后执行：根据要求，悔棋被限制为最多连续两步。

五子棋胜利判定正常工作：在胜利后，双方分别播放胜利与失败音乐，并显示双方总走棋时间。

支持残棋保留现场，在双方开始前可以载入：在载入残局之后，双方可以继续进行游戏。

即，完成了 PPT 上要求的所有基本功能

特色功能

实现了独立稳定运行的中央服务器程序

在完成图形界面的三天内，这一中央服务器程序一直稳定地运行于 xuexiao.dotkrnl.com 这一台位于福建泉州电信网络的 Linux 服务器之上，不曾出现崩溃等情况。并根据对于构造、析构函数的跟踪，以及 token 资源的计数，判定在本地对本地的高负载测试下，以及远端服务器的实际运行测试中，均不存在内存泄露等资源泄漏的情况。

实现了完善可靠的应用层协议

应用层协议完善，不（像他人）依赖于系统缓冲区机制进行分包。基于换行的文本数据传输、心跳检测，稳定而可移植。为应用层协议封装了 LineSocket 及 FiveConnection 类，打包于 dotFiveLibrary 中，共用于服务器及客户端之间。开放的接口，方便二次开发。在近日 CERNET 恶劣的网络，以及北京教育网至福建泉州电信的高延时环境下，可以较为正常地工作。

游戏逻辑判定位于服务器上，外挂成为往事

以坏人假设为原则设计的中央服务器，在恶意输入数据下仍可正确地工作。即便客户端程序出现异常逻辑，或是外挂程序，游戏公平性仍受到服务器的保护。

采用了多线程事件驱动设计，最大限度利用系统计算能力

使用的线程数与 CPU 个数一致，将各个对象的事件合理分布于线程中，在获得多线程高效率的同时，避免了操作系统上下文切换的代价。事件驱动、以及良好的设计使得整个程序中没有锁的存在，进一步使得程序效率得到提升。

支持多用户同时在线进行多场游戏，并支持观战功能

仅需创建四位密码并告诉对手，在连接界面中使用这一密码，即可轻松开始游戏。观众使用同样的密码，即可进入游戏观看双方对战。更支持 Magic Match 功能，只需对战双方同时按下按钮，即可轻松开始游戏。

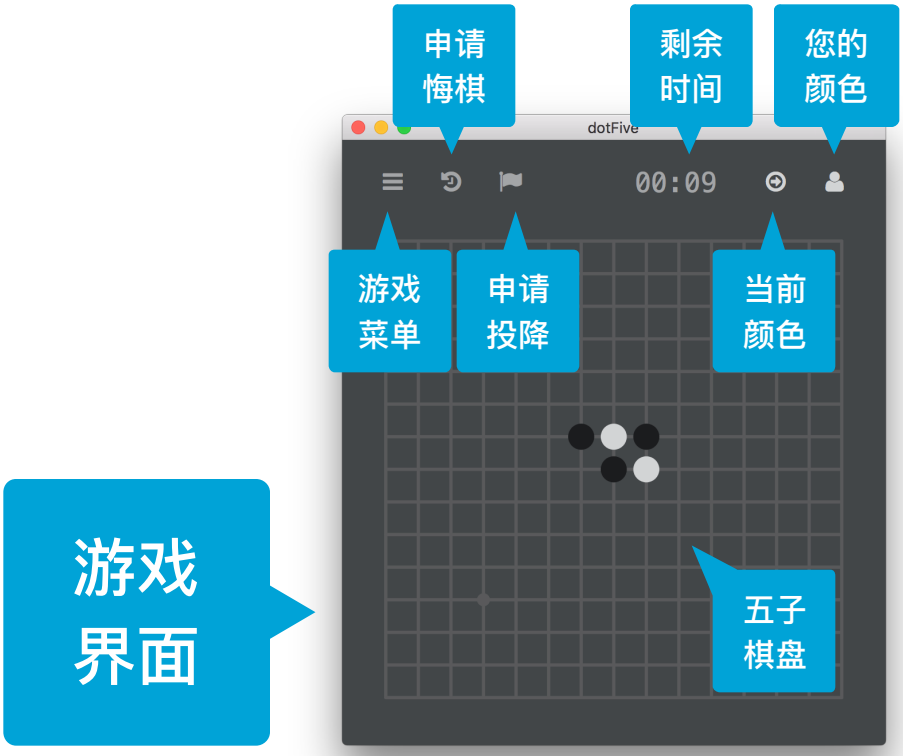
界面展示

开始
界面


dotFive
开始
游戏
加载
存档
切换
颜色
0468
游戏
密码

连接
界面


dotFive
输入密码一致
即可进入游戏
0000


游戏
界面
申请
悔棋
剩余
时间
您的
颜色
dotFive
00:09
游戏
菜单
申请
投降
当前
颜色
五子
棋盘


游戏
菜单
返回
游戏
保存
残局
双方
用时
00:31
00:25
观战
密码
0468

收到
请求

胜利
界面

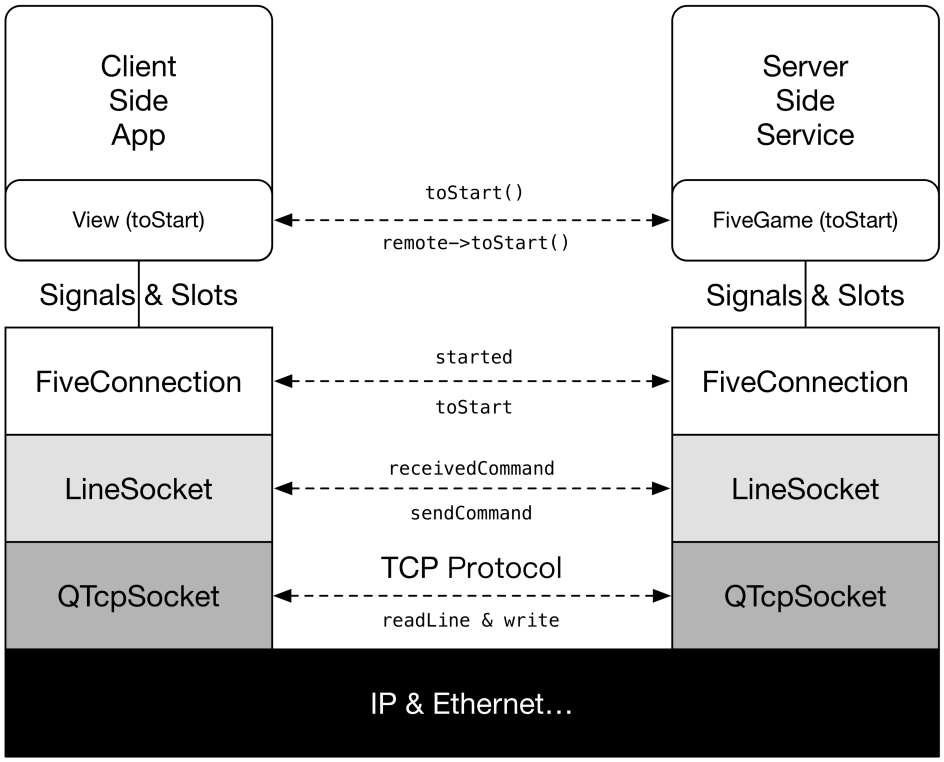
通信框架

我在 TCP 传输层协议之上实现了一套简单的应用层协议 `LineSocket`。

这套协议为基于行的指令系统，格式为 “command `argv[0] argv[1]…\r\n`”。实现了两个 SIGNALs 和两个 SLOTs，分别为接收到指令的 `void receivedCommand(QString command, QStringList argv);` 链接断开的 `void disconnected(void);` 发送指令的 `void sendCommand(QString command, QStringList argv);` 以及断开链接的 `void disconnect(void);`。上述协议利用了 `QTcpSocket` 中的 `canReadLine()` 以及 `readLine()` 完成。

`LineSocket` 同时实现了一个简单的心跳检测系统，实时了解网络状况，并清理坏死 `QTcpSocket`，节约系统资源。并保证在长时间无内容交换的情况下，TCP 网络连接不被 NAT 设备自动切断。

在这之上，我实现了一个简单的五子棋服务器协议 `FiveConnection`。这是一个对于 `LineSocket` 的封装。对于其接收到的数据进行解析，发送对应 SIGNALs。并提供了一系列 SLOTs，进行指令的发送。SIGNALs 和 SLOTs 一一对应，相当于实现了一个简单的远程过程调用（RPC）协议。这一协议被同时用于客户端与服务器中。具体内容请见下一部分。



通信协议

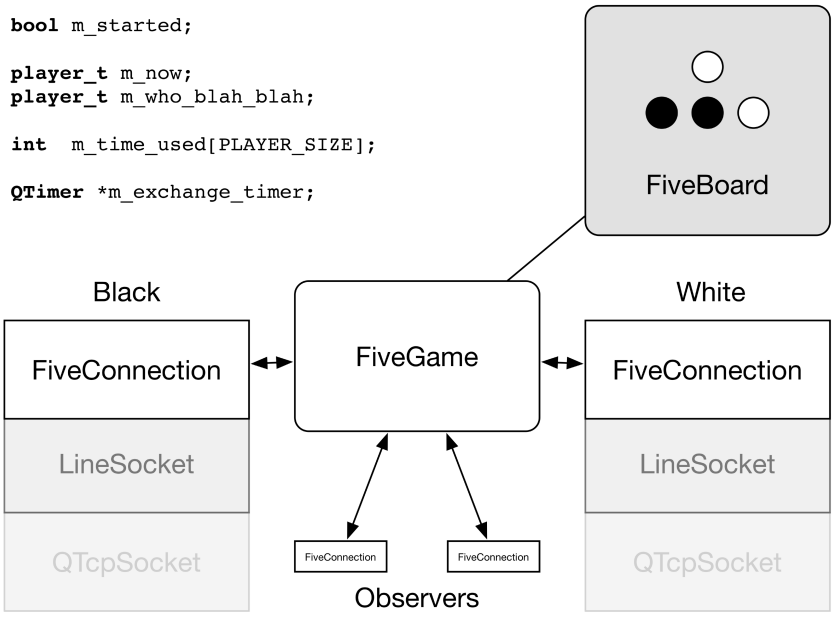
服务器端与客户端共用一套通信协议，其功能如下表。

其中指令为 `FiveConnection` 中包装的 SLOTs 名称。

指令	发往客户端的指令意义	发往服务器的指令意义
`toToken(t)`	已创建/加入游戏，其密码为 t。	加入密码为 t 的游戏。
`toSetColor(c)`	已修改用户颜色为 c。	请求将颜色修改为 c。
`toChangeTurn(c)`	当前可操作用户颜色为 c。	请求让出操作权给 c。
`toStart`	游戏已开始。	请求开始游戏。
`toSync(l,e,c)`	棋盘 l 位置的棋子存在状况为 e，并且其（若存在）颜色为 c。客户端需无条件信任服务器信息，并更新本地显示数据。	请求在 l 指定的位置放置颜色为 c 的棋子，或根据 e 删除。与游戏规则不符的指令，只在游戏未开始的超级管理员权限时有效。
`toRequestUndo`	对手请求执行悔棋操作。	向对手请求悔棋操作。
`toReplyUndo(a)`	对手接受/拒绝了悔棋操作，客户端将会接收到后续的 toSync 信息	接受/拒绝对手的悔棋操作。
`toRequestGiveUp`	对手请求投降。	向对手请求投降。
`toReplyGiveUp(a)`	对手接受/拒绝了投降，客户端将会接收到后续的 toFinish 信息。	接受/拒绝对手的投降。
`toError(e)`	服务器返回了错误。	客户端返回了错误。
`toChangeTime(c,t)`	颜色为 c 的用户共用时 t 毫秒。	不适用。
`toFinish(c)`	游戏结束，胜者为 c。	不适用。
`toCreateToken`	不适用。	创建一个新游戏。创建成功后，将对应密码返回给客户端。
`toCheck(s)`	不适用。	请求检查数据完整性，其校验和为 s。若校验和不一致，服务器将利用指令发送给客户端完整的游戏数据，保持同步的一致性。
`toMagic`	不适用。	请求 Magic Match。

在服务器端，每个已加入游戏的 FiveConnection 都将直接由 FiveGame 接管，一个 FiveGame 可以接入多个 FiveConnection，其中有两个为玩家，其余为观察人。FiveGame 维护了整场游戏的信息，并处理除了加入/创建游戏之外的所有指令信号，向客户端发送游戏逻辑一致的指令信号。

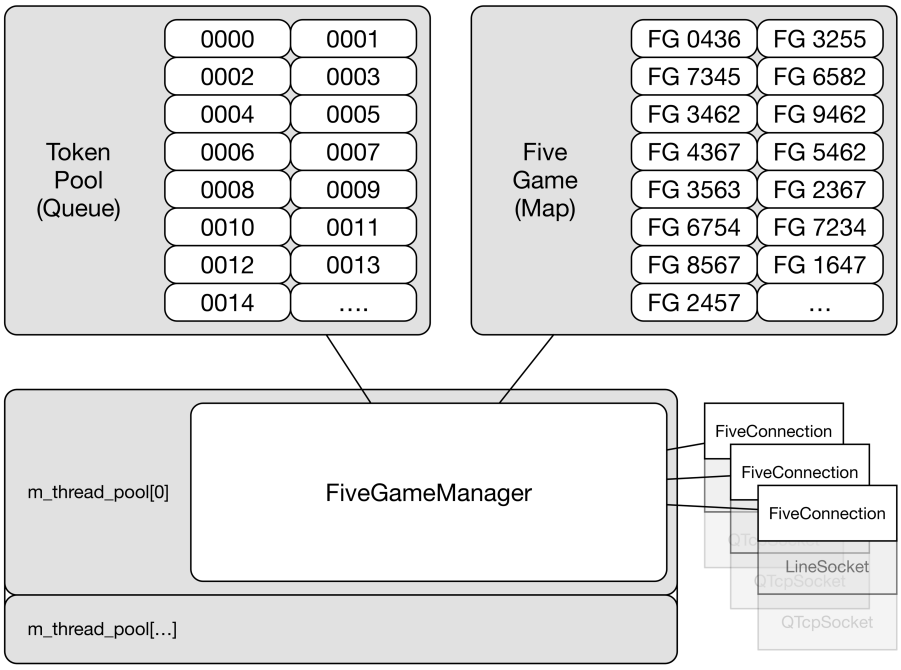
在 FiveConnection 断开时，FiveGame 将释放其对应的对象实例。若此时已没有客户端接入，FiveGame 将发送 gameTerminated 信号，拒绝新的用户加入游戏。这一信号将由后面介绍的 FiveGameManager 接受，并负责释放此游戏的对应资源实例。



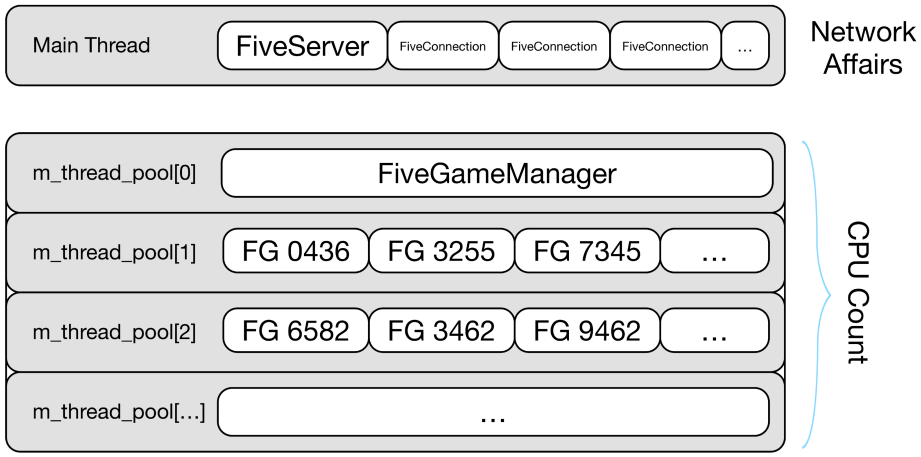
FiveGame 示意图

而 FiveGameManager 则维护了尚未加入 FiveGame 的 FiveConnection，确保其对象资源被正确处理。FiveGameManager 最主要的功能为维护 FiveGame 列表以及可用的密码资源。在申请创建游戏时从密码资源中取出空闲的四位密码，产生新的 FiveGame 对象，令其接管这一 FiveConnection 对象；在申请加入游戏时，从 FiveGame 列表中取出对应的对象接管链接。同时处理 Magic Match 功能所需的信息。

在接收到 FiveGame 的 gameTerminated 信号后，FiveGameManager 将释放对应的对象资源，将对应的密码资源放回资源 pool 中以待使用。



`FiveGameManager` 示意图



线程分配示意图

继承于 QTcpServer，我实现了一个 FiveServer 类，其功能极为简单，监听端口，创建套接字，包装为 FiveConnection 并交予 FiveGameManager 维护即可。

在服务器端程序中，所有代码均为事件驱动编程，不需要多线程支持。但为了使性能最优化，充分利用多核处理器能力（其实是为了装逼求加分），我将上述各个类的对象分布到了 CPU 个数的线程中。对象间的通信完全通过 SIGNALs 和 SLOTs 完成。其中，主线程用于网络通信，其接入连接，接收来自 FiveGameManager 与 FiveGame 的指令，发送给客户端，并接收来自客户端的指令，发给对应的对象处理。线程池中的第一个线程用于 FiveGameManager 维护 token 与 games，其余线程用于各个 FiveGame 的逻辑处理。

客户端利用相同的 Library，完全使用事件驱动方法。由于没有计算压力，不做多线程处理。

在这一架构下，客户端程序完全为 View，只需一一执行来自服务器的指令显示内容，将用户的操作发送给服务器即可。没有值得介绍的架构设计。

收获感想

这一周对 Qt 的学习，使得我很好地练习了利用信号实现多线程事件驱动编程，给了我很好的练习网络编程的机会。我对于 Qt 的多线程实现有了更为深刻的理解：在对象内单线程，在对象外利用信号进行多线程。这样的做法使得 Qt 同时拥有了 Node.js 的无锁高效，也同时有了多线程的资源利用率。并且在多线程环境下，提升了我编写、调试无锁程序的能力。

总的来说，我在这一周的学习中收获很大，也希望再接下去的学习中能够有更大的收获。

感谢老师，以及助教。