**课程编号：B080106000**

**分布式系统导论课程设计**

**《谁说谎》游戏系统的  
技术分析和概要设计**



|  |  |  |  |
| --- | --- | --- | --- |
| **组名** | **WilDCaT** | | |
| **组长(100%)姓名** | **王凌翔** | **组长学号** | **20124862** |
| **副组长(95%)姓名** | **张赢** | **副组长学号** | **20124879** |
| **副组长(95%)姓名** | **佟雨泽** | **副组长学号** | **20124859** |
| **组员(80%)姓名** | **张昕昊** | **组员学号** | **20124901** |
| **组员(80%)姓名** | **张金钰** | **组员学号** | **20124900** |
| **参与人员(75%)姓名和学号(不多于5人)** |  | | |
| **评定成绩** | **100** | **评定人** | **宋 杰** |
| **评定日期** | **2015-2-12** |

**东北大学软件学院**

|  |  |
| --- | --- |
| **评分标准** | **得分** |
| 需求明确：清晰定义分布式系统的技术需求(25%)； | 25 |
| 设计完整：涵盖大部分所学知识点(15%)； | 15 |
| 设计正确：可以完成既定技术需求(30%)； | 30 |
| 自主知识运用：是自己想的还是查阅资料获得的(20%)； | 20 |
| 选题新颖：常规的教科书均可见的还是较为新颖的分布式系统(10%)； | 10 |
| 大段文献抄袭且不给出引用的，对应考核点按零分处理 | |

# 目 录

[目 录 2](#_Toc409812093)

[1. 系统简介 4](#_Toc409812094)

[2. 体系结构 4](#_Toc409812095)

[2.1 技术需求 4](#_Toc409812096)

[2.2 概要设计 4](#_Toc409812097)

[3. 进程 5](#_Toc409812098)

[3.1 技术需求 5](#_Toc409812099)

[(1) 数据的接收 5](#_Toc409812100)

[(2) 控制游戏流程 5](#_Toc409812101)

[3.2 概要设计 6](#_Toc409812102)

[(1) 数据接收 6](#_Toc409812103)

[(2) 控制游戏流程 7](#_Toc409812104)

[4. 通信 8](#_Toc409812105)

[4.1 技术需求 8](#_Toc409812106)

[(1) 客户端、服务器通信协议 8](#_Toc409812107)

[(2) 服务器内部进程间通信 8](#_Toc409812108)

[4.2 概要设计 8](#_Toc409812109)

[(1) 通信协议设计 8](#_Toc409812110)

[(2) 服务器内部进程间通信 9](#_Toc409812111)

[5. 命名 9](#_Toc409812112)

[5.1 技术需求 9](#_Toc409812113)

[(1) 客户端命名 9](#_Toc409812114)

[5.2 概要设计 10](#_Toc409812115)

[(1) 客户端命名 10](#_Toc409812116)

[6. 同步 10](#_Toc409812117)

[6.1 技术需求 10](#_Toc409812118)

[(1) 逻辑时钟 10](#_Toc409812119)

[(2) 互斥 10](#_Toc409812120)

[6.2 概要设计 10](#_Toc409812121)

[(1) 逻辑时钟同步 10](#_Toc409812122)

[(2) 保证进程间互斥 11](#_Toc409812123)

[7. 一致性与复制 11](#_Toc409812124)

[7.1 技术需求 11](#_Toc409812125)

[(1) 增加客户端缓存 11](#_Toc409812126)

[(2) 读写一致性 11](#_Toc409812127)

[7.2 概要设计 11](#_Toc409812128)

[(1) 增加客户端缓存 11](#_Toc409812129)

[(2) 读写一致性 11](#_Toc409812130)

[8. 容错 12](#_Toc409812131)

[8.1 技术需求 12](#_Toc409812132)

[(1) 客户端的断线重连 12](#_Toc409812133)

[(2) 客户端传输异常数据 12](#_Toc409812134)

[8.2 概要设计 12](#_Toc409812135)

[(1) 客户端的断线重连 12](#_Toc409812136)

[(2) 客户端传输异常数据 13](#_Toc409812137)

[9. 总结 13](#_Toc409812138)

[参考文献 13](#_Toc409812139)

# 系统简介

这是一个扑克牌游戏。本系统的目的在于能够实现多个玩家能够一起打牌的场景。服务器端为一个集中式结构。多个客户端连接到该服务器上。服务器内部通过多个connecter进程来维护与客户端之间的连接。由controller进程来控制游戏的流程。在最后的报告中，将附上关于本报告的一份使用C、python编写的服务器、客户端原型，以及一份关于这个游戏的详细规则介绍、程序流程、以及通信协议的说明文档。

# 体系结构

所谓体系结构，即一个系统中各个组件的划分、组织以及组件间的交互方式。其中，软件体系结构是指一个系统中各个软件组件的组织和交互方式；而系统体系结构是指确定软件组件，这些组件的交互方式以及他们的位置，是软件体系结构的一个最终实例[1]。我们设计的“谁说谎”游戏所采用的是系统体系结构中的集中式体系结构。

## 2.1 技术需求

“谁说谎”游戏是一个扑克游戏，支持人数为2到6人。首先我们先来了解一下现实中的游戏基本情况。在现实中，第一步是分牌，即每个用户得到相等数量的扑克牌（如果牌数不能被人数整除，则有的人将会得到多一张的手牌）。之后则进入了出牌，声明，判定和加牌阶段，直到某一玩家将手牌全部出光，则游戏结束。

根据这样的游戏规则，不难发现，该游戏的系统体系结构比较适合使用集中式体系结构。在集中式体系结构中，服务器是一个可以实现多种服务的进程；而客户端是通过像服务器发送请求，并得到服务器的相应反馈的进程。对于“谁说谎”游戏来说，每一个参与游戏的牌手即虚拟为一个客户端，而对于用户对扑克牌的操作的管理和反馈则可以虚拟为一个服务器端，客户端只需向服务器发送请求并得到反馈，而客户端之间并不需要交互。因此，系统采用集中式体系结构。

为了连接的可靠性，系统使用了基于连接的协议。客户端与服务器通过建立IPV4/TCP协议进行连接。

## 2.2 概要设计

Client

Server

Client

Client

**图2-1. 体系结构示例图**

如图2-1所示，为3名玩家同时进行游戏时的体系结构示例图。开始的时候，服务器首先进行初始化。当有用户想要加入时，客户端会像服务器请求连接，然后服务器会对客户端所请求的连接进行相应的处理。这样，客户端与服务器的连接就建立起来了。当参加人数达到要求之后，游戏便开始了。随后在游戏的过程中，服务器会首先为每位玩家进行发牌操作，确定每位用户的应有手牌。当轮到某一位玩家时，服务器会等待该玩家的指令，并对不同的指令进行相应的操作与反馈。

系统使用的是两层体系结构，即只有客户端与服务器，客户端与服务器均有对于应用程序的相关操作。具体组织结构如图2-2所示。

****

**图2-2. 客户端-服务器组织结构示意图**[1]

# 3. 进程

单进程的集中式服务器设计，无法满足本系统的要求。所以需要对服务器上的进程进行设计。

## 3.1 技术需求

### (1) 数据的接收

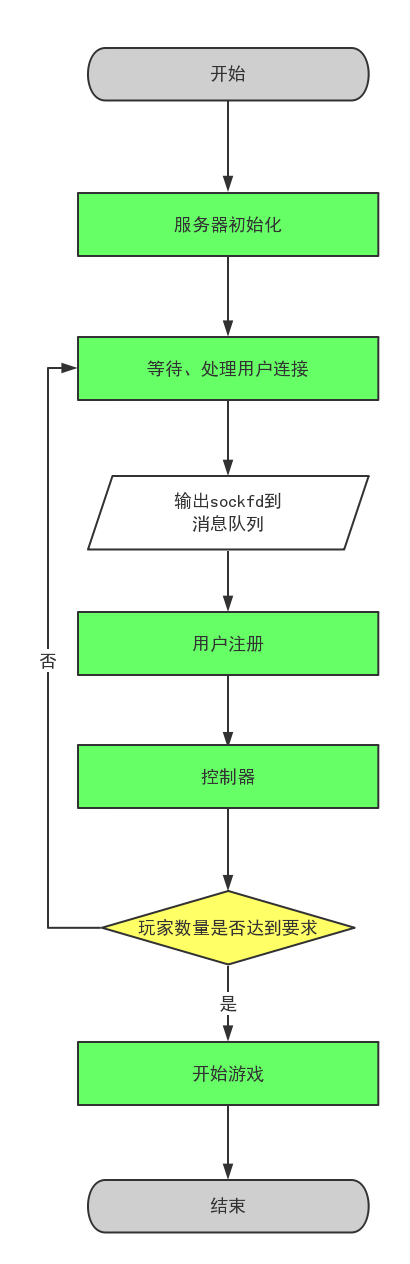
服务器应能接收不同客户端发来的消息，且不互相冲突。

### (2) 控制游戏流程

控制流程应能控制游戏的流程。并且负责游戏的逻辑处理。

## 3.2 概要设计

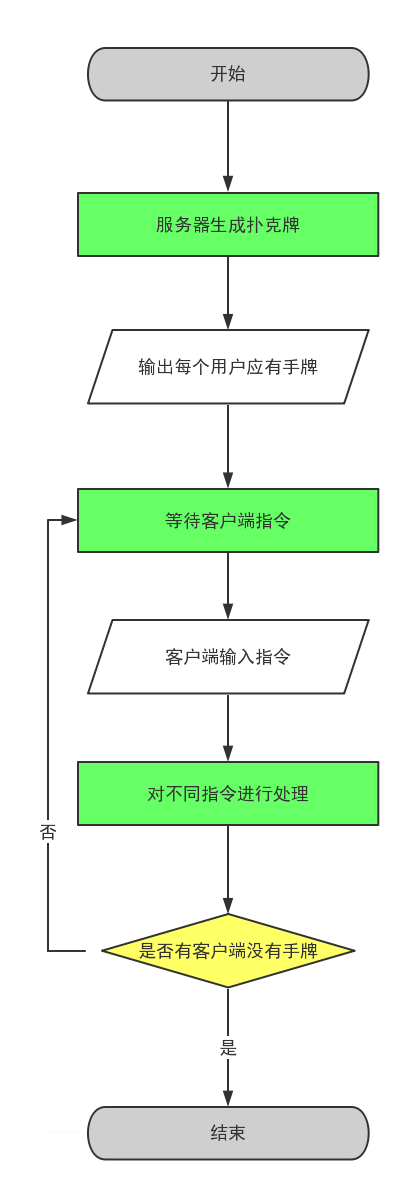
### (1) 数据接收



**图3-1 数据接收的初始化流程**

流程图如上图，每个connecter对应一个客户端，然后将客户端接收到的指令，进行包装，放入消息队列当中。

### (2) 控制游戏流程



**图3-2 游戏流程控制简要流程图**

游戏流程控制如上图所示。服务器为一个有状态的服务器。因为需满足即使在connecter崩溃的情况下，controller仍能正常使用，所以controller应为一个独立的线程或进程来单独控制游戏的流程。

# 4. 通信

客户端与服务器之间需要建立某种通信手段，来进行消息的通信。也就是牌局的进行。而在服务器内部，由于是多进程服务器，进程与进程之间也需要某种通信手段，使得各个进程之间能够成功的交换数据。

## 4.1 技术需求

### (1) 客户端、服务器通信协议

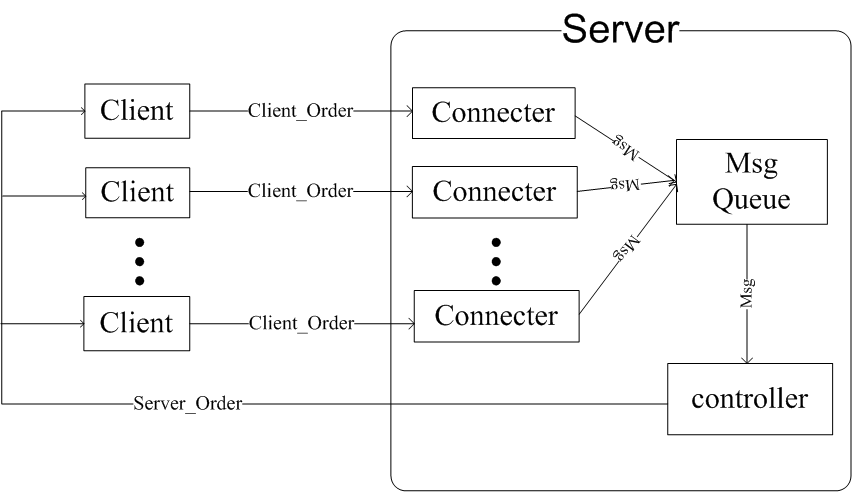
客户端与服务器之间应建立连接，并制定相应的通信协议，使得牌局游戏能够正常进行。

### (2) 服务器内部进程间通信

服务器进程之间应该建立某种渠道，使得服务器内部的各个进程可以进行数据的交换。

## 4.2 概要设计

主要涉及两个方面，一个是通信协议的设计，另外一个是消息队列的设置。



**图4-1 服务器客户端结构图**

### (1) 通信协议设计

1. 扑克牌用一个字符代替，不考虑花色。数字2至9用所对应的字符代替。数字1用字符“A”代替，10用字符“+”代替，11用字符“J”代替，12用字符“Q”代替，13用字符“K”代替。
2. 服务器采用以下语句与客户端进行通信：
3. FORCE [pokers]

说明：服务器发牌给客户端（包括前期手牌和判定失败产生的手牌）

1. TURN

说明：服务器要求客户端进行出牌操作

1. JUDGE

说明：服务器要求客户端进行判定操作

1. 客户端采用以下语句与服务器进行通信：
2. PUT [pokers]

说明：客户端进行出牌

1. CLAIM [pokers]

说明：客户端进行声明

1. SAY TRUE/FALSE

说明：客户端进行判定真/假

1. OVER

说明：客户端声明已没有手牌（游戏开始之后）

### (2) 服务器内部进程间通信

1. Connecter线程（多个）用来维护与客户端之间的连接，并且负责接收由客户端发来的语句。而Controller负责维护游戏的进程，并且负责发送服务器的语句给客户端。所以在服务器建立一个消息队列。因为存在多个Connecter进程以及controller进程同时访问消息队列的情况，所以消息队列Msg\_Queue必须保证进程安全。
2. 存放在消息队列的消息应满足以下结构：

struct msgbuf

{

int mtype; //存储connecter进程的pid

char mtext[120]; //存放client\_order

}msg;

# 5. 命名

用户在牌局当中，会拥有自己的用户名，这需要服务器使用命名技术，对各个客户端进行命名。

## 5.1 技术需求

### (1) 客户端命名

应针对不同客户端，根据用户需求进行命名。

## 5.2 概要设计

基本思路为，由用户输入其所希望用户名，服务器将其记录，最后返回给用户。

### (1) 客户端命名

服务器将维持一个“客户端——用户名”的对照表。在牌局开始前，将允许用户设置自己的玩家名。在牌局开始时，服务器将会把该牌局当中的客户端的对照表发给各个客户端，由各个客户端在输出时完成命名的转换工作。服务器在传输消息时，仍然传输客户端名，在客户端输出给用户时，再进行转换。

# 6. 同步

进程间的同步问题对于分布式系统来说是一个很重要的问题，它包括的内容有物理时钟同步，逻辑时钟同步，互斥等。而对于我们所设计的“谁说谎”游戏，则要保证其中的两点：（1）. 我们不需要物理始终上的同步，而要保证逻辑时钟的同步。在游戏中，每名玩家的回合时间应该固定，而这就需要客户端的逻辑时钟与服务器端同步。

（2）. 保证进程间的互斥。

## 6.1 技术需求

### (1) 逻辑时钟

“谁说谎”游戏与现实中的实际时间并没有关系，所以我们不需要考虑。但是为了防止某些玩家出牌时间过长，每一回合都要应当有时间限制（在原型设计中并未体现）。而这就要求我们必须将客户端与服务器端的逻辑时钟进行同步，否则可能会产生服务器端计时完毕，而客户端并未结束回合的错误结果。在这里，我们应该采用Lamport逻辑时钟。

### (2) 互斥

分布式系统的基础是多线程之间并发和协作[1]，所以在某些情况下会存在进程同时访问相同资源的情况，由此而造成系统的崩溃或信息的不一致，这种时候我们就需要引入方法来实现进程之间的互斥。在“谁说谎”游戏中，Connecter线程（多个）用来维护与客户端之间的连接，并且负责接收由客户端发来的语句。而Controller负责维护游戏的进程，并且负责发送服务器的语句给客户端。所以就存在着多个进程同时访问服务器端消息队列的情况，这就可能导致程序的崩溃。我们需要保证消息队列的安全。

## 概要设计

### (1) 逻辑时钟同步

在服务器端要将指令发送给客户端之前，服务器端Pi执行Ci←Ci+1。

当Pi发送消息m给客户端的时候，把m的时间戳t(m)设置为Ci。

在接受消息m的时候，客户端Pj调整自己的局部计数器为Cj←max{Cj, t(m)}。

### (2) 保证进程间互斥

在对消息队列的读写中，引入管程，对各个进程访问消息队列进行控制，防止出现“脏”数据的情况。

# 7. 一致性与复制

客户端与服务器的数据需要保证一致性，只有这样，牌局才会有条不紊地进行。

## 7.1 技术需求

### (1) 增加客户端缓存

增加客户端缓存主要考虑到客户端可以加快读取数据和上传数据的速度。减少用户等待时间，进而优化用户体验。

### (2) 读写一致性

为防止出现牌局异常情况。服务器也需要对每个客户端的手牌进行存储。所以需要每次出牌后，客户端所记录手牌与服务器所记录手牌进行同步。避免客户端作弊以及牌局状态异常。

## 7.2 概要设计

结合上方一致性与复制。

### (1) 增加客户端缓存

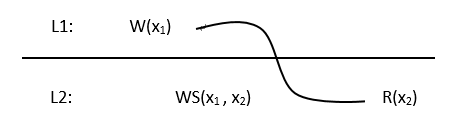
当用户要查看当前手牌时，将直接返回记录在客户端上的手牌。而不必从服务器上请求手牌。提高响应速度，也降低服务器的访问压力。

### (2) 读写一致性

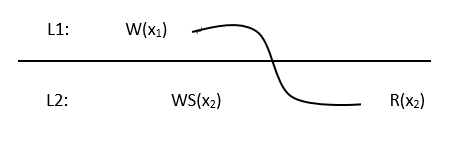
所谓的“读写一致性”是指一个进程对数据项x执行一次写操作的结果总是会被该进程对x执行的后续读操作看见。

举个例子，下图表示了一个提供写读一致性的数据存储，并且一致性是通过进程P执行的最后一次写操作确定的。对图的解释如下：

在图7-1中，进程P执行了一次写操作W(x1)，然后在一个不同的本地副本处执行了一次读操作。写读一致性保证，写操作的结果可以被所有后续的读操作看到。这一过程表示为WS（x1；x2），它表示W(x1)是WS(x2)的一部分。与之相对照，在图7-2中，W(x1)不在WS(x2)之中，这意味着进程P执行的前一个写操作的结果还没有传播到。



**图7-1 提供写读一致性的数据存储**



**图7-2 不提供写读一致性的数据存储**

在本系统中，为保证客户端与服务器手牌的一致性。客户端需在发送指令过后，发送自己此时客户端所存手牌，与服务器所存手牌进行同步。需强调的是，此时所指的服务器所存手牌，是已经经过指令所对应的运算处理过后的手牌。同步时以服务器的手牌为基准，客户端的缓存中，将存储该次服务器处理过后的手牌。

# 8. 容错

基于联网的情况下，难免客户端或服务器会有一方断线。在客户端断线或者传送异常数据的时候，就需要服务器对此进行一些容错处理。在此，我们假设服务器是稳定的，不会受到异常的干扰，而导致服务器出现异常状况。

## 8.1 技术需求

### (1) 客户端的断线重连

在客户端由于某种原因断线的情况下，服务器应该支持客户端进行断线重连，并且牌局当中的状态不应该被改变。

### (2) 客户端传输异常数据

客户端在某些情况下，可能会传输一些不符合协议的数据。服务器应该要能对这些不符合协议的数据进行处理。以维护牌局的正常进行。

## 8.2 概要设计

### (1) 客户端的断线重连

每个牌局服务器将生成一个session\_id。并在这个session\_id中，记录当前牌局的所有状态。而当客户端发送数据时，需要在数据包中携带这个session\_id，表明自己处于哪一个牌局当中。这样，即使客户端短暂的离线，也能保证在重连之后服务器能找到客户端当前所在的牌局。

### (2) 客户端传输异常数据

客户端传输异常数据包括本身数据包在物理链路中，因为种种原因而产生的差错，以及由于用户输入错误数据而产生的异常错误。前者不在本文档的讨论范围之内，不予考虑。现考虑用户输入异常数据。服务器在接受到异常数据之后，发送一个数据包给客户端，要求客户端输入正常数据。此时牌局暂停，等待用户输入正常数据。如果用户超出规定出牌时间，服务器将出牌的权限转交给下一个用户。

# 9. 总结

“谁说谎”游戏是一个考验玩家心理的扑克游戏，也是我们小组成员在课余时间的消遣之一。虽然现在网络游戏平台内的扑克游戏很多，但是却并未见到该游戏的身影，这也是我们设计这款游戏的初衷。该游戏与普通扑克游戏有所不同：普通的扑克游戏，如斗地主等，只是轮流出牌，当谁最先将牌出完游戏即结束。而“谁说谎”游戏在出牌者每次将牌出完并声明之后会随即进入一个判定阶段和一个加牌阶段，这就对游戏系统的设计提出了更高的要求。

针对游戏的机制、规则以及相应特点，结合了我们“分布式系统导论”课上学到的知识，我们完成了对这个游戏的系统设计，并初步实现了该游戏的原型。由于小组成员对游戏的熟悉程度较高，我们的技术需求能够清晰完整的提出。对于每一部分的技术方法，经过对参考文献的查阅，我们往往能够选择出与该游戏最契合的一个，并准确地将之与游戏结合起来。在设计的同时我们也进行了代码上的实现，这让我们避免了纸上谈兵的尴尬，在实践中产生了对分布式系统更深层次的理解。

当然，目前我们的游戏系统还只是一个原型，所涉及到的一些技术需求还不能完全实现。本次课程设计之后，我们还将不断完善它。

# 参考文献

1. Andrew S.Tanenbaum，Maarten Van Steen，辛春生，陈宗斌等译 《分布式系统原理与范型》（第二版）[M].北京：清华大学出版社，2008年