山东大学（威海）

SHANDONG UNIVERSITY , WEIHAI

毕业论文（设计）

设计(论文)题目　　L0L 实时对战数据安卓发布软件

姓　　名：　高楚阳

学　　号：　201200800568

学　　院：　机电与信息工程学院

专　　业：　通信工程

年　　级　　2012

指导教师：　梁成辉

2016 年　　5 月　　21 日

摘　　要

针对英雄联盟这款游戏，市面上流行的对战数据发布软件多运行与 PC 端，且不能针对某个用户监控其实时对战数据，对于这款游戏数据监控软件的移动化、实时化需求强烈，Android 平台智能终端的迅速发展，为英雄联盟对战数据发布软件的移动化、实时化提供了保障。

本文针对英雄联盟这款游戏对战数据的实时化、移动化发布需求，提出了一种实时数据监控软件的解决方案。该方案分为服务端通信框架、数据管理、安卓客户端数据展示三个核心模块。服务端采用 Mina 通信框架，支持高并发多客户端访问。安卓客户端采用安卓图表引擎开发图形绘制部分，并利用子线程进行网络通讯，使用 Handler 类与主线程交互，更新界面，减少了线程污染。

关　键　词

Android 开发，英雄联盟，MINA

Abstract

For the League of legends game, the gaming data monitoring software is commonly running on PC terminal, however, it’s inconvenient for user to get realtime gaming data. Therefore the strong demand for mobile realtime monitoring software and rapid development of the Android platform smart terminal provide a technical support for the mobility of the gaming data monitoring software.

Combining with mobile monitoring software mobility and real-time needs, this paper put forward a gaming data mobile monitoring software system solution. it is divided into 3 modules including server communication framework, data management, Android client. The server is implemented based on an asynchronous network communication framework named MINA. Android client use achartengine to draw chart and use class handler to update chart .

KEYWORDS

league of legends, Android, MINA

目　　录

一、前言

本设计的目的是客户端接收服务器发送的实时对战数据并以特定方式展示，提供给分析师以便于对数据进一步挖掘。

本设计面向的是电竞俱乐部的教练与数据分析师，以便于他们能在实时数据层面，定量的掌握本局对战的信息，指定下一步的战术战略打法，从而赢得比赛的胜利。可以说使用了该设计，实时洞察敌我双方实力不是梦。

本设计可以在对战实时发生时，及时的获取第一手信息，并开展分析，以便于制定下一步的对策。通常来说，实力未知的对手常常会给新人玩家带来一份未知的恐惧。但在 LOL 实时对战数据发布软件的支持下，用户却能在游戏过程中准确查看到双方队伍的资源掌握情况，并能借此估算出双方队伍的实力，从而达到合理分配各线英雄的目的。

对于 LOL 电竞俱乐部而言，充分了解每一位对手的综合实力是必要的战备工作。而大部分战败的普通用户，也同样希望了解战胜自己的对手是一位怎样的强者。而在该设计中，用户便可以通过选择召唤师昵称的方式，轻松搜索到任意玩家的实时对战数据。另一方面，完整官方数据的实时更新让自身的实时对战数据可以一目了然，为用户提供一个理性分析自己水平，找出不足，提高自我的平台。同时也给当前对战用户一份理论上的数据，为下一步的战略布局，打法提供一份客观的数据支持，有效避免了依据“想当然”、“我能反杀”等主观意识而得到的错误战术，提高了本局对战的胜率，为中国队能在世界比赛上取得好成绩提供了一种合理的努力方向。

实际上，如若能适当的使用好这些实时数据带来的优势，也将在一定程度上提升用户本身的胜率。以团战为例，用户在 LOL 实时对战数据发布软件中查询到自己在本次对战中已经有全场最高的被控制时间，则可针对性的选择出保命装备，并合理控制自己的走位，从而对为下一步的团战做好充分的准备。而分析对手的上单、中单等习惯，同样可以在布局中取得极大的优势，令整个游戏过程不用“心算”也能第一时间掌握各项动态，再也不用担心错过团战的最佳时机了。

LOL 是一款多人联机在线竞技游戏，它有着完善的输出系统、天赋系统、符文系统、装备系统、英雄系统、资源系统，目前，依托其数据系统，国内的安卓数据发布软件有 TGP、盒子、大脚等等，这些软件可以从英雄、战绩、战斗力、隐藏分等方面全面的了解自己对手，比如时下由腾讯公司开发的 TGP 软件，可以看到近 30 天的表现，得到 30 天内召唤师峡谷匹配赛和排位赛使用英雄统计（不含逃跑对局），并求出最近 30 场的平均 KDA 值、胜率。对于已经结束的匹配赛与排位赛，可以进行对局分析。但是，这些安卓软件都存在着一个不足，那就是仅仅是提供历史对战数据，只能通过历史的对战数据来分析对手的习惯与打法，制定策略，如果本局对战对手采用了新的套路，实际表现与以往数据大相径庭，那么根据以往数据制定的策略将毫无用武之地，失去了针对性。当下的国内的直播软件虽然可以通过视频转播比赛的实时对战状况，却缺少了具体数据的定量展示，只有定性的直播视频展示，凭借视频中得到的主观定性信息无法对场上局势作出客观定量的描述，从而无法提出进行精确定量的战术指导方案，从而失去本局比赛的主动权，将胜利的机会主动交于别人手中。

二、服务器通信框架设计

本实时数据发布软件采用服务器-客户端结构，在服务器进行实时对战数据管理工作。

（一）基于 socket 的多用户通信框架

方案一为采用基于 socket 的多用户通信框架实现实时对战数据的推送和多客户端的对接。

Java 被创造出来以后,它的多线程能力、连接互联网能力、和强大的跨平台能力使惊艳了新时代的每一个人，Java 技术和互联网的强强联手，使得 Java 和互联网都迅速发芽生长，展现了强大的活力。在互联网技术的不断演进的大背景下 ，不同平台的系统之间的信息交互变得越来越重要 ,在这个时候 ,互联网行为的一个非常令人注意的领域———多用户信息传递，也给互联网上运行的应用程序带来更多的挑战。Java 语言具有强大的平台无关性,它的网络编程功能和多线程机制成为实现网络多用户通信的首要考虑目标[1]。

1.LOL 实时对战数据发布软件中的套接字通讯机制

LOL 实时对战数据发布软件服务器端通过以 Socket 来实现的 TCP 协议进行与安卓客户端的数据传输，本服务器中的一个 Socket 由一个 IP 地址和和一个端口号（本设计中采用 9898 端口）唯一确定。使用 Socket 对实时数据服务器与安卓客户端建立连接。实时数据发布软件服务器端一直处于监听状态,当实时数据发布软件安卓客户端向服务器所监听的端口（本设计中采用 9898 端口）发起连接请求时，实时数据服务器端程序就返回一个包含客户端 Socket 信息 Socket 对象，这样就实现了一个与安卓客户端的专门虚拟连接通道。安卓客户端程序向 Socket 中添加请求内容， 实时数据服务器端程序通过对应的 Socket 收到请求内容,然后解析其中的数据后返回给安卓客户端程序。安卓客户端与服务器端的信息交互完成,就关闭这个虚拟连接通道。通讯机制见图 1。

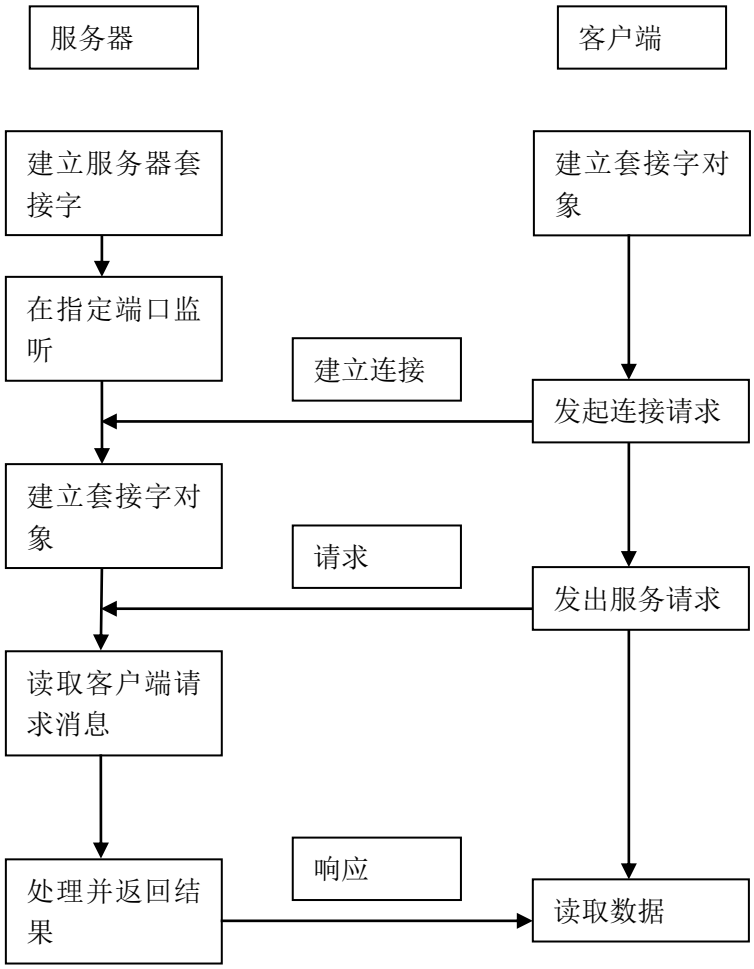

图 1 原生 Socket 通信机制

当 LOL 实时数据发布安卓客户端通过 9898 端口与实时数据发布服务器连接时,该服务器有唯一确定并且可以追踪监视和安卓客户端机进行信息交互的 Socket 对象。实时数据发布服务器使用这种监视追踪的方法，使用 9898 端口与多个安卓客户端进行通信传输数据。所有连接到服务端的安卓客户端的 Socket 对象都被实时数据发布服务器端所拥有。LOL 实时数据发布服务端可以分辨来自不同的安卓客户端的数据请求内容，它通过不同的 Socket 对象分别对不同的安卓客户端的请求内容给出不同的推送内容。LOL 实时数据发布服务器端使用 JAVA 的多线程机制开辟多个线程,通过这些线程处理不同安卓客户端的网络 I/O 操作,从而实现对多个安卓客户端的并发实时对战信息推送。

LOL 实时数据发布软件的服务端和安卓客户端采用基于 Java 的 Socket 的通信方式，使用本方案写出的实时对战数据订阅与推送代码可以像文件 I/O 一样，实时数据服务器端从 Socket 中读取安卓客户端订阅的实时对战数据和向 Socket 写实时对战数据,非常容易地使用网络资源。LOL 实时数据发布软件的服务器端的底层机制是实时数据服务器主函数中对端口侦听进行循环,使用阻塞函数响应安卓客户端的连接请求。

2.实时对战数据推送服务器端的处理原理

首先建立绑定到 9898 端口上的套接字: SocketServer socketServer = new SocketServer(9898);实时对战数据服务端使用 9898 端口循环监听安卓客户端发起的连接请求 : socket = serverSocket.accept();安卓客户端发起请求时,accept()函数返回一个保存有安卓客户端信息的 Socket 对象,完成安卓客户端与对战数据服务端的连接。对战数据服务端接受连接之后 ,使用 获取的客户端 Socket 对象中的 getInputStream()和 getOutputStream()方法,建立文件 I/O 数据流对象 InputStreamReader 和 OutputStreamWriter 与安卓客户端通信,这样就实现对战数据服务端与安卓客户端的数据交互。

3.实时对战数据安卓客户端并发连接的处理原理

当一个安卓客户端发起向对战数据服务端的连接请求后,对战服务器开启此次与安卓客户端会话，但会因为需要等待客户端的后续请求而执行阻塞操作,为了实现对战服务器同时与多个安卓数据发布客户端通信,对战数据服务器端使用 JAVA 的多线程处理机制,数据服务器循环监听 9898 端口,每次接收客户端的发起的连接请求后,创建一个 socket 对象 ,并建立一个新的线程,把保存有安卓客户端信息的 Socket 对象交给新建立的线程, 使用这个特定的新线程完成与这个数据发布安卓客户端的通信。服务端 main 函数则继续执行循环结构，监听 9898 端口,直到 main 函数终止，服务器关闭。(如图 2)

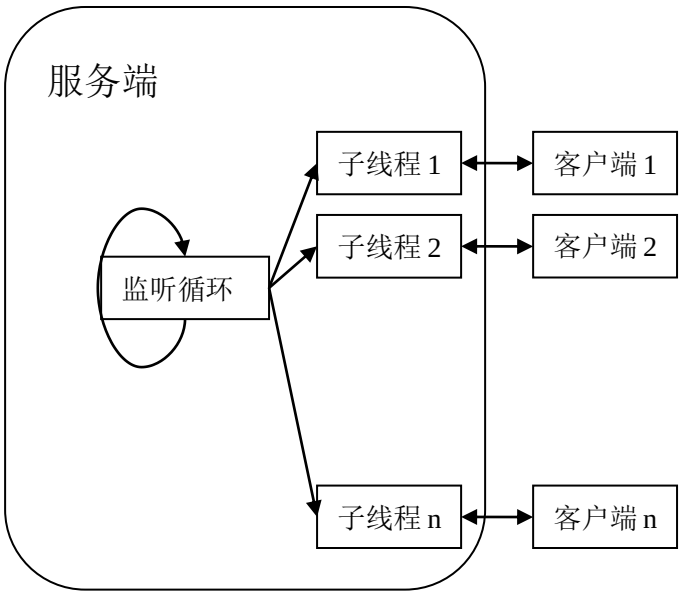

图 2 原生 Socket 多客户端连接实现机制框图

（二）使用 Apache Mina Server 网络通信框架

方案二：采用 Apache Mina Server 网络通信框架对实时对战数据进行推送,并读取安卓客户端发送过来的订阅数据

随着使用本软件监控实时对战数据的用户的不断增加,会对实时对战数据服务器端程序提出越来越高的要求。 一个服务器端同时连接几百上千个安卓客户端， 也会变得司空见惯。这就对服务器端程序的高性能提出了挑战，因此， 在本设计中需要使用一种高效的网络 I/O 底层，来更容易地开发出高性能的网络

应用程序。Java NIO 和 Mina 的出现， 给本设计提供了一种解决方案。 Mina 采取异步 I/O 和事件驱动机制，有很高的效率和性能。结合 Java NIO,所以本设计可以采用 Mina 更容易地开发出高性能的实时对战数据服务端程序。

1.采用 Mina 框架的实时对战数据服务端实现原理

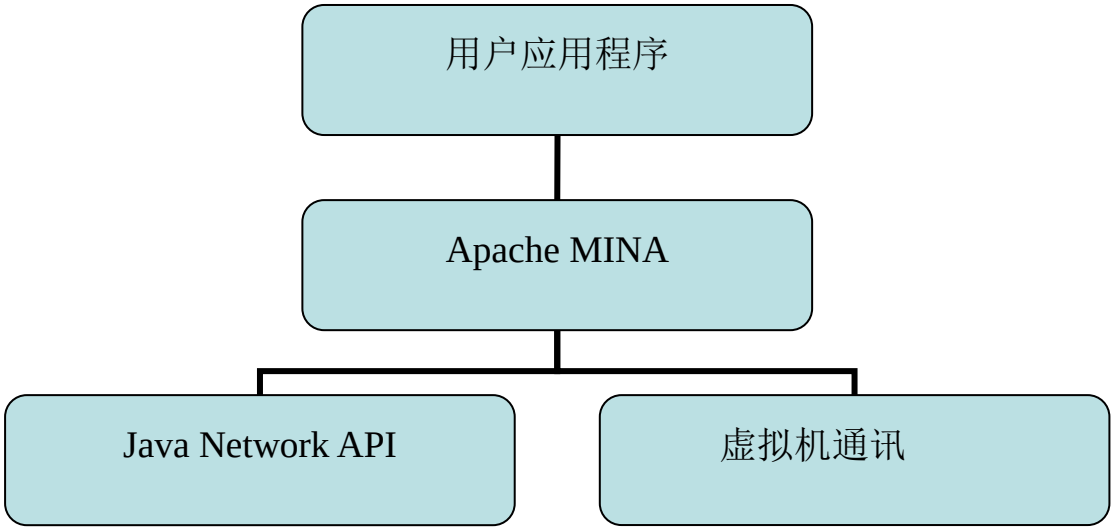

图 3 Mina 的架构图

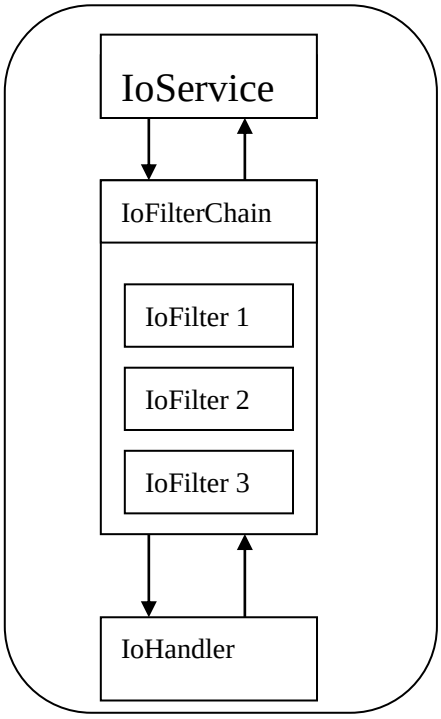

图 4 Mina 详细架构图

实时对战服务端程序使用模块链中的 IoService 类作为服务端应用程序的入口。代码为：NioSocketAcceptor acceptor = new NioSocketAcceptor();

2.服务端对于 IO 流的编码解码处理

服务端程序使用 I/O Filter Chain 对于输入与输出的 io 流进行过滤。代码：acceptor.getFilterChain().addLast("codec",newProtocolCodecFilter(new MyTextLineFactory()));

其中使用自定义方式对 io 流数据进行编解码，通过 MyTextLineFactory 实现。

新建一个类 MyTextLineFactory，继承接口 ProtocolCodecFactory,重写其中的 getDecoder 和 getEncoder 方法，分别对应了解码和编码的方式。具体实现方法为建立一个名为 MyTextLineEncoder 的类，继承接口 ProtocolEncoder，实现编码；建立一个名为 MyTextLineDecoder 的类，继承接口 ProtocolDecoder，实现解码。在 getDecoder 中返回 MyTextLineDecoder 的一个对象,在 getEncoder 中返回 MyTextLineEncode 的一个对象。

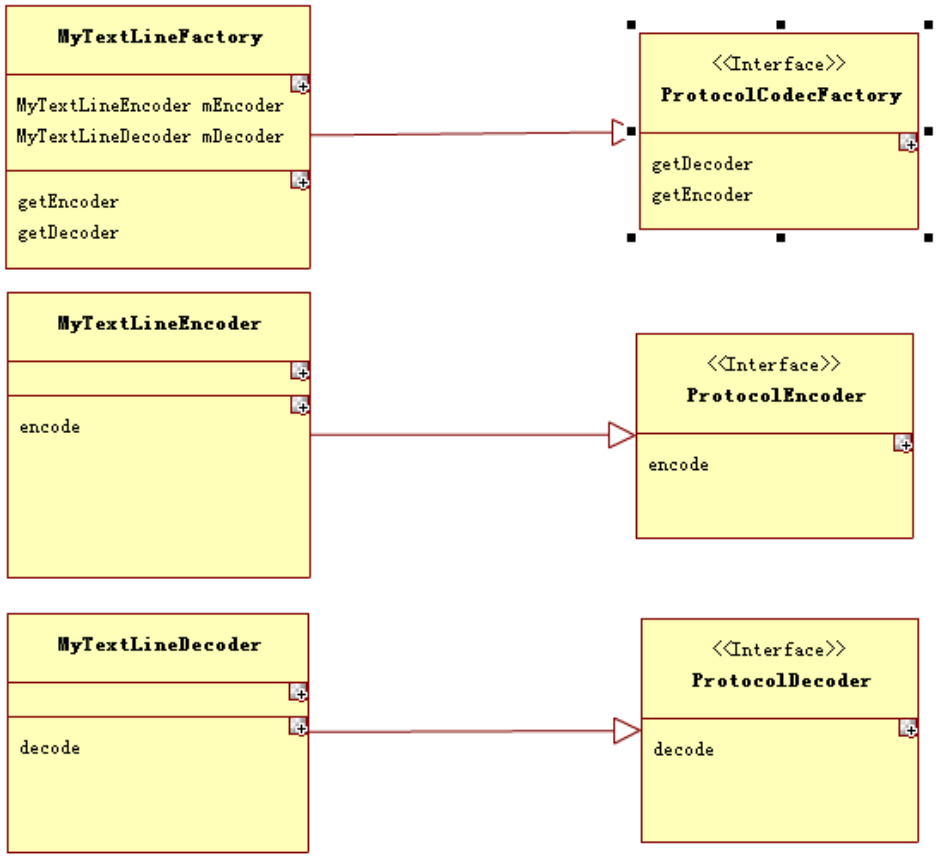

图 5 过滤器类图

（1）服务端编码方式

由于发送出去的内容可能是一个对象，但在网络层仅允许字节的传输，所以发送的消息必须先经过编码成为字节串,重写 MyTextLineEncoder 中的 Protoco--lEncoder 接口函数，实现 message 对象到字节的转换，方法为：首先对要发送的 Object 类型的对象 message 进行判断，如果它不是 String 类型，则将其转换为 String 类型;然后对转换后的 message 对象进行转码操作，使用 IoBuffer 对

象的 allocate 方法开辟一段长度为 message.length 的内存，使用 ioBuffer 的 setAutoExpand 方法实现内存的动态扩展，使用 Iobuffer 的 putString 方法将转换为字符串的 message 按照系统默认的 UTF-8 编码方式进行编码。使用 flip 函数表示转换结束，使用 ProtocolEncoderOutput 类型的一个对象中的 write 方法把转码后的 ioBuffer 对象发送出去。

（2）服务端解码方式

由于在网络层数据的传输是以字节流的形式，重写 MyTextLineDecoder 中的 ProtocolDecoder 函数对接受到的数据进行解码，恢复成为一个字符串。具体解码方式为：首先读取接受到的内容，然后将收到的字节转换为字符串。具体方法为：首先使用一个整型变量 startPosition 记录开始读取字节流的位置，然后开启一个 while 循环，使用 iobuffer 中的 hasRemaining 方法判断字节流中是否还有数据，决定是否继续循环，循环体的内容为，使用 Iobuffer 的 get 方法每次读取字节流中的一个字节 b，如果我们读取到的字节为\n，则表示读取结束，然后进行结束处理。结束处理时，先使用 iobuffer 的 position 方法记录下当前读取到的位置 currentPosition，然后使用 iobuffer 的 limit 方法记录当前的总长度，然后进行截取操作，将一行开始位置与结束位置之间的所有数据截取出来。使用 iobuffer 中的 position 与 limit 方法进行重定向，使用 slice 方法进行截取操作。截取出来的 iobuffer 对象存入字节数组中，然后将这个字节数组转换成 String 类型，方法为使用 String 构造函数，将获得的字节数组写入 Stirng 构造函数的初始化列表中。最后使用 ProtocolDecoderOutput 类型的一个对象中的 write 方法把转码后的 ioBuffer 对象发送出去。截取后再用 position 与 limit 方法对于位置进行还原，将 positon 从 startPosition 还原至 currentPosition 保证下一次截取的正确性，避免陷入死循环。

3.服务端业务处理模块

实时数据服务端使用 IoHandler 类作为业务处理模块。在业务处理类中不需要去关心实际的通信细节，只管处理客户端传输过来的信息即可。为了简化 Handler 类，Mina 提供了 IoHandlerAdapter 类，此类仅仅是实现了 IoHandler 接口，因此，只要在具体的业务逻辑中重写相应的方法即可。Iohandler 方法如下图。

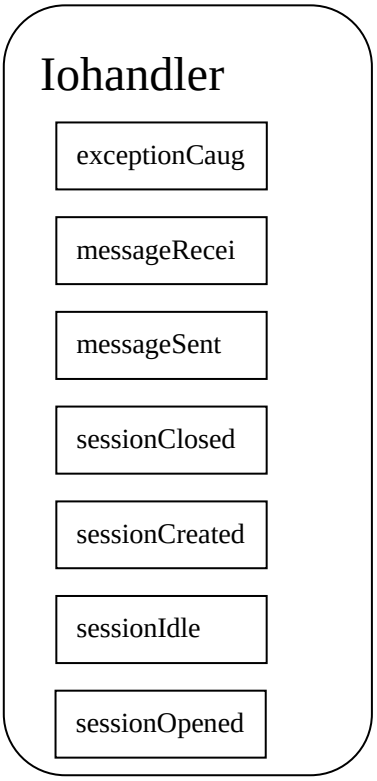

图 6 IoHandler 封装方法

IoHandler 封装了来自客户端不同事件的处理， 如果对某个事件感兴趣，可以实现相应的方法，当该事件发生时，IoHandler 中的方法就会被触发执行。IoHandler 总共有 7 个方法对应 7 个事件：

（a）void exceptionCaught(IoSession session, Throwable cause)
有异常发生时被触发。
（b）void messageReceived(IoSession session, Object message)
有消息到达时被触发，message 代表接收到的消息。
（c）void messageSent(IoSession session, Object message)
发送消息时时被触发，即在调用 IoSession.write()时被触发，message 代表将要发送的消息。
（d）void sessionClosed(IoSession session)
当连接关闭时被触发，即 Session 终止时被触发。
（e）void sessionCreated(IoSession session)
当创建一个新连接时被触发，即当开始一个新的 Session 时被触发。
（f）void sessionIdle(IoSession session, IdleStatus status)
当连接空闲时被触发。使用 IoSessionConfig 中的 setIdleTime(IdleStatus status, int idleTime)方法可以设置 session 的空闲时间。如果该 Session 的空闲时间超过设置的值，该方法被触发，可以通过 session.getIdleCount(status)来获取 sessionIdle 被触发的次数。
（g）void sessionOpened(IoSession session)
当打开一个连接时被触发。在目前的实现中，好像 sessionOpened 和 sessionCreated 没有太大区别，sessionCreated 在 sessionOpened 之前被触发。

IoHandler 是一个接口，一般情况没有必要直接实现该接口的每一个方法。MINA 提供了一个 IoHandlerAdapter 类，该类实现了 IoHandler 要求的方法，但是都没有做任何处理。实时对战数据服务端采用扩展 IoHandlerAdapterHandler 的方式，实现了自己的 Iohandler，并重写了其中的事件方法。[2]

通过重写 IoHandlerAdapter 中的 messageReceived 方法实现了对客户端订阅信息的解析。具体解析方法为：

订阅数据分为三段，第一段表示请求内容是否为当前在线情况，第二段表示订阅单种类，第三段表示请求的具体那个人的 ID 号。

第二段数据的对应情况如下表：

表 1 订阅单代号表

订阅单内容	订阅单代号
敌我实力从敌我实时经济能力	0
实时视野控制情况	1
史诗级中立生物实时控制情况	2
资源掌控实时数据	3
敌我团队贡献从敌方我方实时参团率	4
输出率	5
金钱比	6
承受伤害率	7
实时 KDA 值	8
补兵&中立生物实时数量	9
击杀史诗级中立生物实时个数	a
敌我双方实时英雄数据通过实时死亡分布	b
实时击杀分布	c
获得金钱值	d
实时 KDA 值	e
实时补兵值	f
实时出装	g
关键技能实时使用情况	h

根据请求内容的不同，服务器端会将不同的数据打包发送给客户端，这样就实现了特定数据的推送。

根据上述两个方案的比较，我们发现采用基于 socket 的多用户通信框架方案在处理多客户端连接时速度相对较慢，因为其采用了阻塞式 I/O 进行网络通信，如果有 100 个客户端进行并发连接就要建立 100 个线程进行处理，增加了服务端的负担，降低了通信速度。而 MINA 框架方案在处理多客户端连接时表现较好，有利于对高并发连接的处理，并且 MINA 框架将各个模块独立开来，提高了内聚性，降低了耦合性，有利于后续的二次开发，故本设计采用方案二。

（三）服务端数据管理

1.对战数据模型建立

服务端建立了一个 data 类作为数据模型，用来模拟真实用户在对战时产生的数据。用户在真实对战中产生的数据主要分为团队位置、团队角色、击杀死亡情况、造成或承受伤害、其他（治疗量、金钱等等）五个方面。

击杀数据包括：击杀英雄数量，角色死亡数，助攻数，最高连杀数。

造成伤害数据包括：对英雄造成伤害，对英雄造成物理伤害，对英雄造成魔法伤害。

承受伤害数据包括：回复生命值，承受总伤害，承受物理伤害。承受魔法伤害。

其他数据包括:获得金钱数量，摧毁防御塔数量，击杀小兵数量，击杀中立生物数量、视野控制情况、击杀大龙数量、击杀小龙数量。

团队位置数据分为：上单、中单、射手、辅助、打野。

团队角色包括：坦克、物理输出核心、魔法输出核心、治疗型辅助。

2.对战数据仿真器运行方式

服务端使用 HashMap 管理所有用户的数据对象，并使用定时器每秒对所有用户的数据进行一次改变。

对战数据仿真器模拟数据的方法为：

所有用户的金钱数每秒加一，击杀小兵数每十秒加一，击杀中立生物数每三分钟加一，魔法和物理伤害量每秒钟加一百，回复生命数每秒加三。

团队角色为坦克的用户每秒承受一个随机的一千以内的物理伤害和随机数为一千以内的魔法伤害。承受总伤害为承受物理伤害加上承受魔法伤害。

团队角色为物理输出核心的用户每秒钟增加一个随机的 500 以内的对英雄物理伤害。角色为魔法输出核心的用户每秒钟增加一个随机数为 200 以内的对英雄魔法伤害。角色为治疗型辅助的用户每秒钟增加一个随机数为 200 以内的对治疗量。

团队位置为上单的用户，每秒补兵数加一，金钱增加二十以内的随机数，位置为中单的用户每秒金钱增加三十以内的随机数，每秒补兵数加一，位置为射手的用户，每秒金钱增加三十以内的随机数，每秒补兵数加二。位置为打野的用户，每秒金钱增加十以内随机数，并且每秒击杀中立生物数加一。

三、安卓客户端设计

（一）安卓客户端的 MVC 模式

客户端使用列表控件 ListView 作为界面的主要形式。作为显示当前在线的具体用户的容器，ListView 控件采用 MVC 模式将前端显示与后期数据进行分离。也就是说，ListView 控件在装载数据时并不是直接使用 ListView 类的 add 或类似的方法添加数据，而是指定了一个 Adapter 对象。该对象相当于 MVC 模式中的 C（Controller，控制器）。ListView 相当于 MVC 模式中的 V（View，视图），用于显示数据。为 ListView 提供数据的 List 或数组相当于 MVC 模式中的 M（Model）模型。[3]

在 ListView 控件中通过控制器（Adapter 对象获得需要显示的数据）。在创建 Adapter 对象时需要指定要显示的数据（List 或数组对象），因此要显示的数据与 ListView 之间通过 Adapter 对象进行连接，同时又互相独立。也就是说，ListView 只知道显示的数据来自 Adapter，并不知道这些数是来自 List 还是数组。对于数据来说，只知道将这些数据添加到 Adapter 对象中，并不知道这些数据会被用于 ListView 控件或其他控件。

安卓实时数据客户端采用 ArrayList 作为数据模型，一方面管理接收到的当前在线用户数据，另一方面管理不同类别的用户订阅单，并将它们加入控制器 Adapter 对象中，与 ListView 连接并显示在界面上。

（二）安卓客户端的网络通信模块与线程间通信

1.线程间通信

实时数据发布客户端使用主线程（也就是 UI 线程）管理界面中的 ListView 控件，进行事件分发，并完成实时数据图形绘制工作。使用子线程进行与服务器的网络通信工作，使用 handler 实现子线程与主线程的通信工作，以便于更新数据。

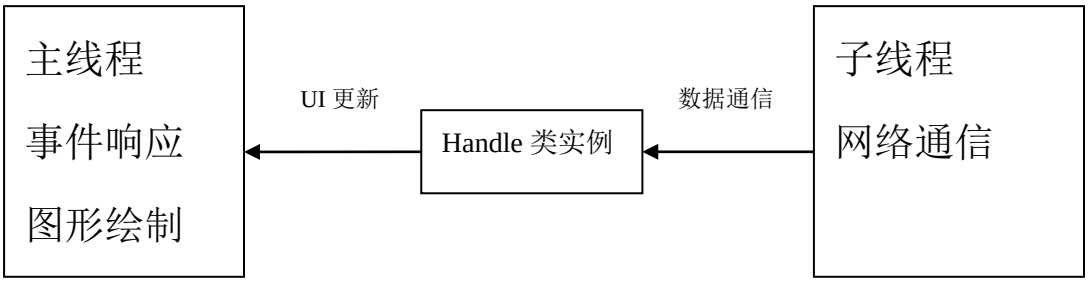

图 7 子线程与主线程通信示意图

当客户端应用程序启动时，Android 首先会开启一个主线程（也就是 UI 线程），主线程为管理界面中的 UI 控件，进行事件分发，比如说，单击本客户端中 ListView 的一个 item，Android 会分发事件到该 item 上，来响应你的单击操作。如果此时需要一个耗时的操作，例如：联网读取数据，或者读取本地较大的一个文件的时候，就不能把这些操作放在主线程中，如果将网络 I/O 操作放在主线程

中的话，界面会出现假死现象，如果 5 秒钟还没有完成的话，会收到 Android 系统的一个错误提示“强制关闭“。此时客户端会出现闪退现象。这个时候我们需要把这些耗时的操作，放在一个子线程中，因为子线程进行网络通信接收到数据后涉及到 UI 更新，而 Android 主线程是线程不安全的，也就是说,UI 只能在主线程中更新，子线程中操作是危险的。所以,实时数据发布客户端采用 Handler 来解决这个复杂的问题，使用运行在主线程中(UI 线程中)的 Handler，它与子线程可以通过 Message 对象来传递数据，这个时候，Handler 就承担着接受子线程传过来的(子线程用 sedMessage 方法传递)Message 对象(里面包含数据)，把这些消息放入主线程消息队列中，配合主线程进行更新 UI。

2.网络通信模块

使用原生套接字类进行客户端网络通信，具体方法为：

在 activity 类中声明一个私有的 Socket 类型的成员变量,并另起一个线程，对其初始化。使用 Socket(InetAddress address, int port)方法创建一个流套接字并将其连接到指定 IP 地址的指定端口号。对于 PC 端的安卓模拟器使用 IP 地址 10.0.0.2，对于手机端的安卓软件，使用主机真正的 IP 地址，二者均采用连接至 9898 端口。

客户端网络通信编码流程：使用 socket.getOutputStream()方法得到此套接字的输出流，并用这个输出流创建使用默认字符编码（UTF-8）的 OutputStreamWriter。OutputStreamWriter 是字符流通向字节流的桥梁：可使用指定的 charset 将要写入流中的字符编码成字节。本设计中直接使用默认的字符集，与服务端匹配。为了获得最高效率，客户端再将获得的 OutputStreamWriter 包装到 BufferedWriter 中，使用 BufferedWriter(Writer out)方法创建一个使用默认大小输出缓冲区的缓冲字符输出流。将子线程通过网络通信接收到的文本写入字符输出流,缓冲各个字符，从而提供单个字符、数组和字符串的高效写入，以避免频繁调用转换器。最后根据现有的 BufferedWriter 创建一个 PrintWriter，用来向服务器发送订阅单。

客户端网络通信解码流程：使用 socket.getOutputStream()方法得到此套接字的输入流，利用此输入流创建一个使用默认字符集的 InputStreamReader。InputStreamReader 是字节流通向字符流的桥梁,它读取字节并将其按照默认解码方式（UTF-8）解码为字符，为了达到最高效率，客户端在 BufferedReader 内包装 InputStreamReader，从子线程网络通信字符输入流中读取文本，缓冲各个字符,从而实现字符、数组和行的高效读取。使用 BufferedReader 中的 readLine 方法读取服务器发送过来的一条信息。

进入实时图形绘制界面后，子线程会循环读取从服务器发送过来的数据，服务器发送的消息头包含了发送数据的 ID 号，用来标示发送的是哪个用户的实时对战数据，客户端使用一个布尔类型的变量来控制禁止或允许数据的接收，当退出图形绘制 activity 时，在 activity 的 OnDestroy 方法中将变量置位，禁止在后台接受数据。另外服务器关闭，也会导致客户端停止接受数据。

子线程在进行网络 I/O 操作前先创建出 message 类的一个实例，在收到服务器数据包后，将 ID 号存入 message 的整型变量 arg0 中，将具体数据提取后存入 message 的 Object 类对象 Obj 中，然后使用 handler.sendMessage 方法将这条消息发给主线程。主线程收到这条消息后执行 updatechart 方法，更新图表，从而实现了实时绘制图形，用图表的方式显示服务器实时数据。

（三）客户端图形绘制

1.绘制方法介绍

实时数据发布客户端使用实时条形图，饼状图，折线图三种方式对敌我实力，团队贡献，英雄数据三个方面进行立体化的展示。

采用安卓图表引擎 AChartEngine 进行不同种类动态图表的绘制，以绘制折线图为例简要介绍代码思路：

首先在 Activity 的 OnCreate 方法中对图表进行初始化，使用 ChartFactory 工厂类，通过此类来构建不同类型的图表容器控件，用来呈现不同的类型的图表对象。

在 Activity 类中新建三个方法 getDataset()、getRenderer()、updateChart（）分别用来获得当前图表的数据集容器、渲染器容器、设置图表参数、配合子线程更新图表。

getDataset 的实现方式为，首先建立一个 XYMultipleSeriesDataset 类型的数据集容器，用来存放多条曲线的数据集合。然后创建若干 XYSeries 类型的数据集，用来存放曲线数据，并初始化每一条曲线的名称，最后把所有曲线的数据集加入数据集容器中。

getRenderer 的实现方式为，首先建立一个 XYMultipleSeriesRenderer 类型的渲染器容器，用来初始化坐标系，网格，标题等，还用来存放多条曲线的渲染器。然后创建若干 XYSeriesRenderer 类型的渲染器，用来存放存放曲线的参数，比如线条颜色，描点大小，线型等，并初始化每一条曲线的名称，最后把所有曲线的渲染器加入渲染器容器中。

updateChart 的实现方式为，首先使用 removeSeries 方法移除数据集中旧的点集,然后在数据集中添加从服务器获得的新的点集，将新点集放入数据集中后，使用 invalidate 函数更新视图。

2.客户端展示内容介绍

该设计客户端主要实时展示敌我实力对比，敌我双方团队贡献率，敌我双方英雄数据三个方面的对战数据。

其中，敌我实力从敌我实时经济能力、实时视野控制情况、史诗级中立生物实时控制情况、资源掌控实时数据四个方面进行立体化的展示，用户可以依据此

客观数据制定下一步的战术。以一局实际的比赛为例，比赛进行到 30 分钟时，用户在 LOL 实时对战数据发布软件中查询现在我方的经济实力不如敌方，但是视野控制能力大大超过对面，那么下一步可以重点考虑使用埋伏战术，抢先占据敌方未知区域的视野，通过埋伏的方式团灭对方，从而可以取得本局比赛的胜利。部分展示图表如下：

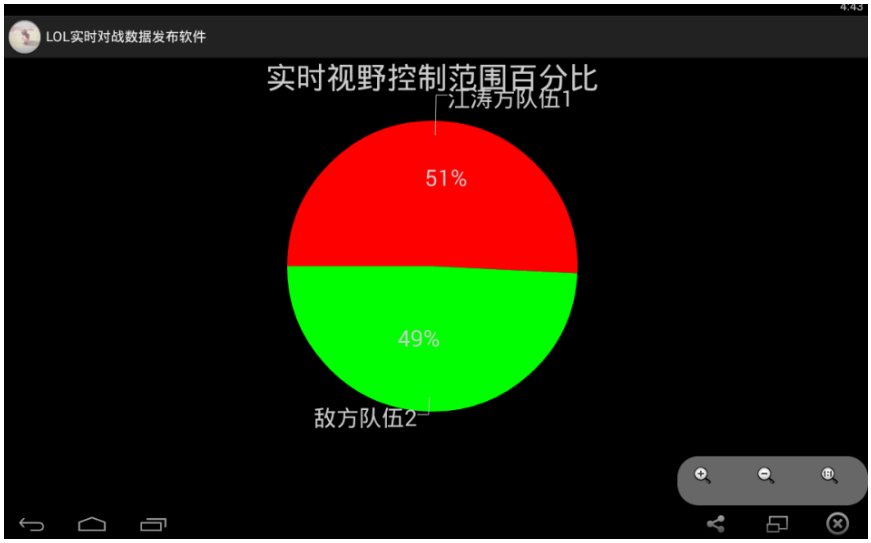

图 8 实时视野控制情况

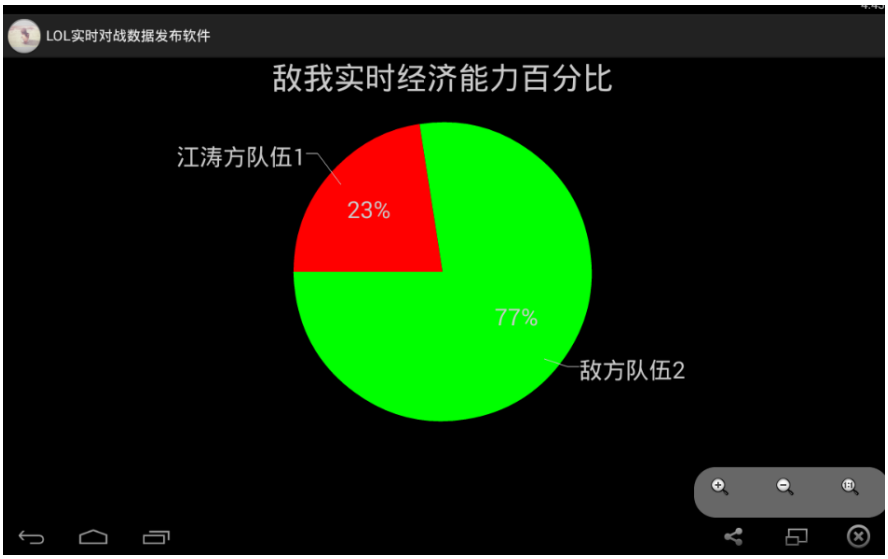

图 9 实时经济能力对比

敌我团队贡献从敌方我方实时参团率、输出率、金钱比、承受伤害率、实时 KDA 值、补兵&中立生物实时数量、击杀史诗级中立生物实时个数等几个方面全面的展示当前时刻每个人对于整个队伍的实时贡献多少，从而为团战、资源抢夺提供一个清晰且最优的思路。以一局排位赛为例，比赛进行到 40 分钟时，往往一局团战的胜负就可以一锤定音，决定整局比赛的胜利，此时用户在 LOL 实时对战数据发布软件中观察到现在对面的中单与打野英雄，团队的贡献率最高并且输出伤害全场最高，那么下一步可以重点考虑优先击杀这两个英雄，并做出有针对性的防御装备，下一场团战中指定准确的击杀方案，顺利的首先解决掉敌方的核

心灵魂人物，从而可以取得此次团战的胜利，奠定了比赛胜利的基础。实际界面展示如下：

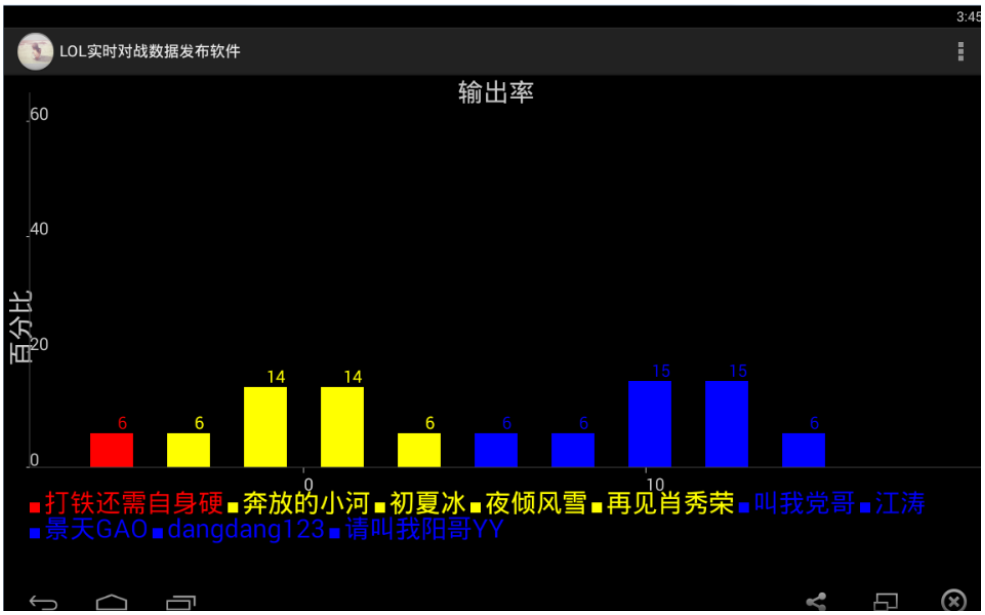

图 10 输出率

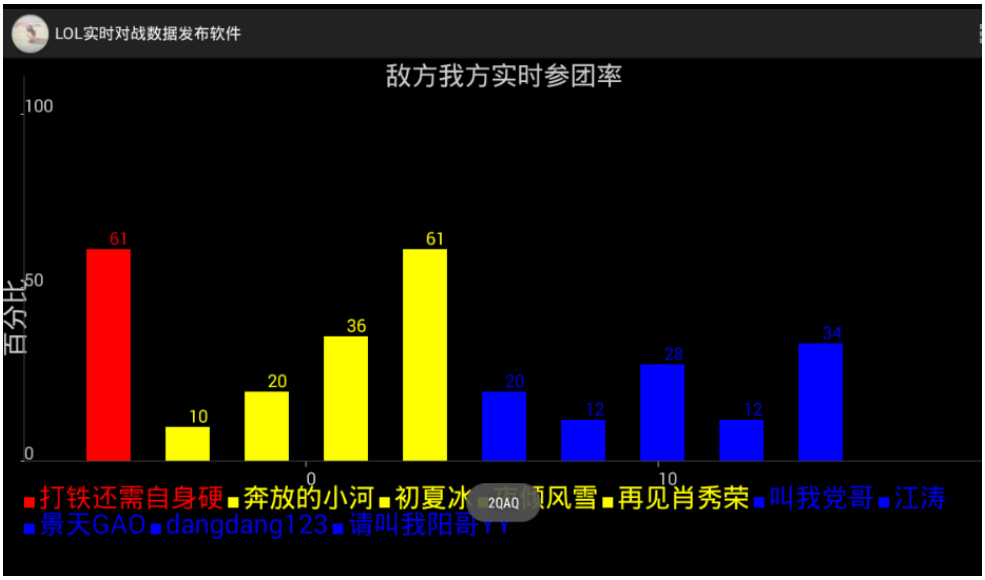

图 11 参团率

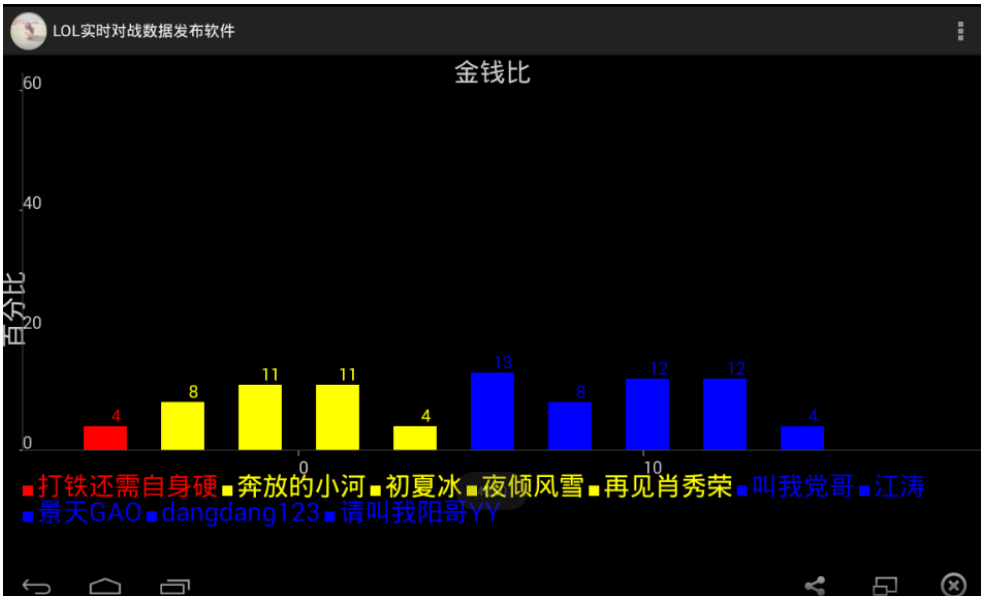

图 12 金钱分配

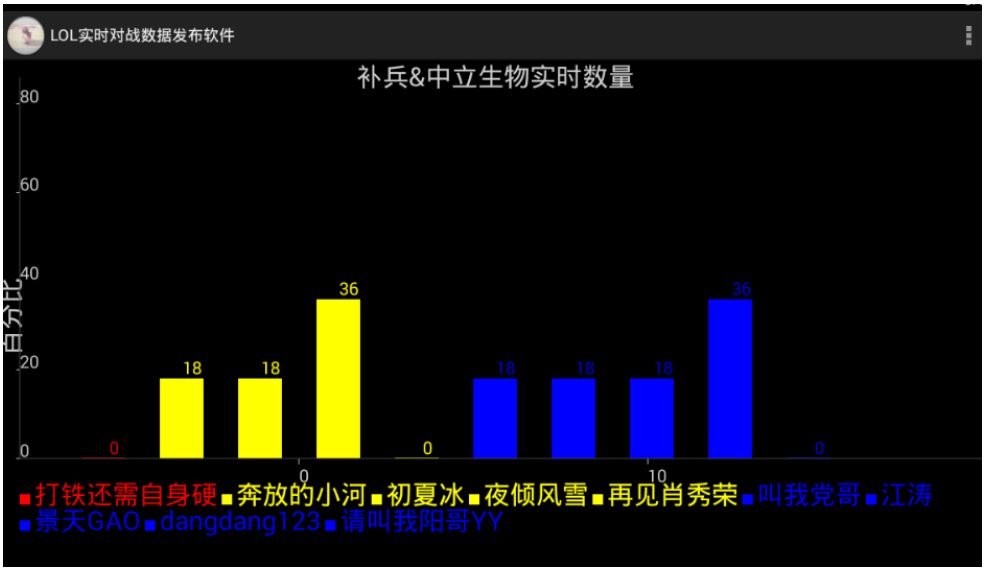

图 13 击杀生物数量

敌我双方实时英雄数据通过实时死亡分布、实时击杀分布、获得金钱值、实时 KDA 值、实时补兵值、实时出装、关键技能实时使用情况等几个方面展示。通过查看本局比赛的实时英雄数据可以针对敌方英雄特点制定有效的针对策略，从而限制敌方队伍核心人物的发挥，降低敌方队伍核心人物的团队贡献率，提高本局比赛的胜率。以一局比赛为例，比赛进行到 10 分钟时，此时比赛刚刚开始，需要寻找一个突破口，迅速打开局面建立优势，此时用户在 LOL 实时对战数据发布软件中观察到现在对面的法系输出英雄的两个关键性保命技能都在冷却状态，那么下一步可以重点考虑对该英雄指定一个有针对性的击杀方案，通过这一次击杀顺利打开了僵局，为此次比赛建立了开局良好的优势。部分界面展示如下：

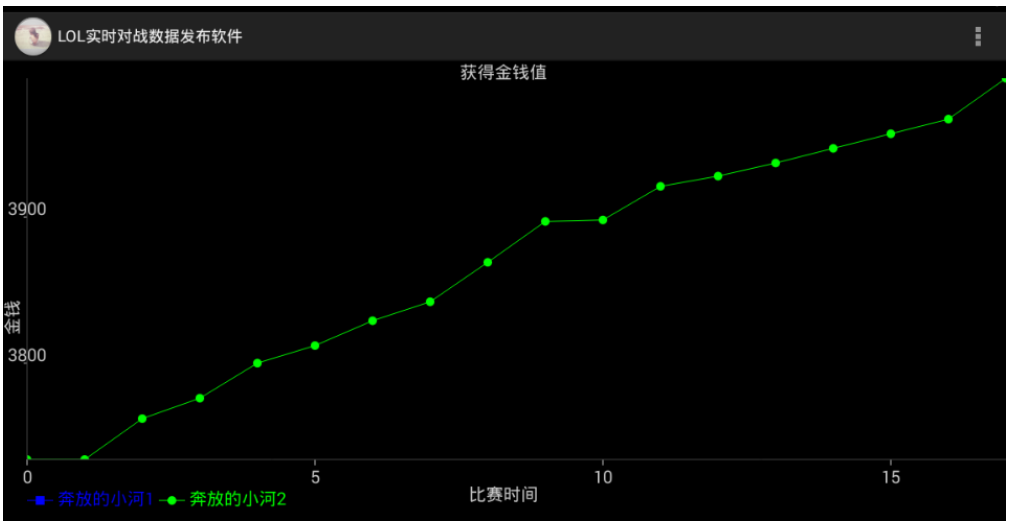

图 14 实时获得金钱值

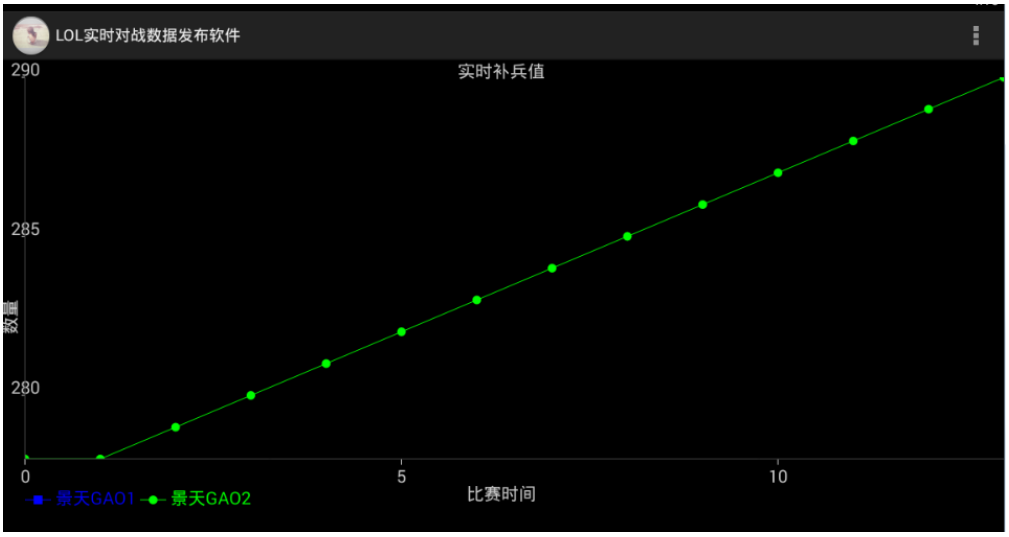

图 15 实时补兵数量

3.软件测试部分

客户端可以顺利取得服务端在线用户姓名，各个 Activity 之间跳转并无 Bug。实际客户端运行效果图如下：

图 16 客户端界面测试

服务端接收订阅数据，推送实时对战数据运行界面如下：

图 17 服务端收发数据测试

四、总结与展望

本设计以实时数据发布为核心功能，有力的弥补了国内同类软件的缺陷，解决了以往软件无法发布实时对战信息的问题，在信息的时效性范围内近最大限度的提供给用户本场对战的定量化数据，从而帮助用户优化下一步的策略。

本设计根据用户对 LOL 实时对战数据的需求，有针对性、有目的性地将用户所需的队伍实力对比、资源分配、团队实时贡献率等信息主动送达用户。其基本工作流程是：

首先服务器推送给客户端当前在线玩家及索引号，客户端获取信息并解析完毕后告知用户可以读取的在线玩家目录信息。

然后用户选择需要订阅的实时对战数据内容，其中包括指定查看玩家、所需的特定对战数据类别等相关内容，并提交给服务器。

最后服务器按用户订阅单收集相关对战数据，并由服务器打包推送给客户端，客户端获取信息并解析完毕后告知用户可读取信息。其内容包括两类：

第一类是只将有关对战数据的目录或索引通知发送给用户，由用户根据对战数据通知再进一步查询相应的对战数据信息。

第二类是直接将所需对战数据中的数据本身打包发送给用户。

本文虽然经过认真的设计，基本达到了预期的目标，取得了一定的成果，但是仍然有着很多未来可以改进的地方和新增的功能。

首先可以通过加入安全登录认证过程有效防止了软件被盗用的可能。保障了数据和文件只能被授予了权限的用户使用和查看，使得可以对于不同级别权限的用户查看不同的文件和数据内容。

还可以增加多场比赛同时监控的功能，实现实时监控分析多个队伍的数据。

最后可以考虑进行跨平台 HTML5 的 web 式开发，能够让软件轻易的在不同平台中运行，降低跨平台的开发成本和周期。

参考文献

[1] 童　铭 ,李志蜀. 基于 Socket 的多用户通信框架及实现. 四川大学学报，2006，43（3）：703-705

[2] 杨铁军 徐和飞 黄 琳. 一种基于 MINA 框架的数据通信平台设计. 微计算机信息，2009，25（11-3）：22-24

[3] 李宁. Android 开发完全讲义. 北京：中国、水利水电出版社，2012 年 4 月. 160～163

谢　　辞

时光总是短暂的，转眼间四年大学时光就要匆匆结束，我得到了许许多多来着学校、老师、同学的各种帮助，以至于每念至此都心生感激。

值此论文完成之际，首先要衷心感谢梁成辉老师，他在我的毕业设计阶段给予了太多悉心的指导，正是在一次又一次的提醒、检查、讨论中我才有了一点一滴的进步。他的严谨治学、忘我工作的态度都深深的影响着我，成为我前进的动力。

感谢初阳，贺宇千，郎浮东、党相卫等同学，正是因为你们的陪伴与支持，我才能由想出这个点子，到一点点完成毕业设计，克服一个又一个的困难。

感谢我最亲爱的父母，感谢你们含辛茹苦的抚养和给予的无私的爱，是你们在一直支持着我、鼓励着我、包容着我，是你们总是在我遇到各种困难时助我度过难关，我用尽此生也难以为报。

感谢学校给我一个学习的环境，让我度过了大学四年，学习了新的知识，有了更宽广的视野。

高楚阳

2016 年于山大威海