山东大学（威海）
SHANDONG UNIVERSITY , WEIHAI

Graduation thesis (Design)

(machine translation English version)

Design (Thesis) topic	real-time battle data Android publishing software

full name:	Gao chuyang
Student No.:	201200800568
Major:	electrical engineering
Grade	2012
Instructor:	Liang Chenghui

May 21, 2016

Abstract

For the game of hero alliance, most of the popular battle data release software on the market runs on the PC side, and can not monitor the real-time battle data for a user. There is a strong demand for the mobility and real-time of the game data monitoring software, and the rapid development of intelligent terminals on the Android platform has contributed to the mobility of the battle data release software of hero alliance Real time provides guarantee.

This paper presents a solution of real-time data monitoring software for the real-time and mobile release of the battle data of hero alliance. The scheme is divided into three core modules: server communication framework, data management and Android client data display. The server adopts Mina communication framework to support high concurrent multi client access. The Android client uses the Android chart engine to develop the graphics drawing part, uses the sub thread for network communication, uses the handler class to interact with the main

thread, updates the interface, and reduces thread pollution.

key word

Android development, hero alliance, 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

catalogue

1、 Foreword

The purpose of this design is that the client receives the real-time battle data sent by the server, displays it in a specific way, and provides it to the analyst for further data mining.

This design is for the coaches and data analysts of the e-sports club, so that they can quantitatively grasp the information of the game at the real-time data level, and specify the next tactical and strategic play, so as to win the game. It can be said that using this design, real-time insight into the strength of both the enemy and ourselves is not a dream.

This design can timely obtain the first-hand information and carry out analysis when the battle occurs in real time, so as to formulate the next countermeasures. Generally speaking, opponents with unknown strength often bring unknown fear to new players. However, with the support of lol real-time battle data release software, users can accurately view the resource mastery of both teams in the process of the game, and estimate the strength of both teams, so as to achieve the purpose of rational allocation of heroes in each line.

For lol E-sports club, it is necessary to fully understand the comprehensive strength of each opponent. Most of the defeated ordinary users also want to know what kind of strong opponent they are. In this design, the user can easily search the real-time battle data of any player by selecting the nickname of the summoner. On the other hand, the real-time update of complete official data makes its own real-time battle data clear at a glance, providing users with a platform to rationally analyze their own level, find out deficiencies and improve themselves.

At the same time, it also gives the current users a theoretical data, provides an objective data support for the next strategic layout and playing method, effectively avoids the wrong tactics based on subjective consciousness such as "take it for granted" and "I can fight back", improves the winning rate of the game, and provides a reasonable direction for the Chinese team to achieve good results in the world competition.

In fact, if we can properly use the advantages brought by these real-time data, it will also improve the winning rate of users to a certain extent. Taking the group war as an example, if the user inquires in the lol real-time battle data release software that he has the highest controlled time in this battle, he can select life-saving equipment and reasonably control his walking position, so as to make full preparations for the next group war. By analyzing the opponent's habits of placing orders and winning orders, we can also obtain great advantages in the layout, so that the whole game process can master various dynamics at the

first time without "mental calculation", and there is no need to worry about missing the best time of group war.

Lol is a multiplayer online competitive game. It has perfect output system, talent system, rune system, equipment system, hero system and resource system. At present, relying on its data system, domestic android data publishing software includes TGP, box, big foot, etc. these software can comprehensively understand their opponents in terms of heroes, achievements, combat effectiveness and hidden points, For example, the TGP software developed by Tencent can see the performance in recent 30 days, get the hero statistics used in the summoner Canyon match and qualifying within 30 days (excluding the escape game), and calculate the average kDa value and winning rate of the last 30 games. For the matches and qualifying matches that have ended, match analysis can be carried out. However, there is a deficiency in these Android Software, that is, they only provide historical battle data. They can only analyze the opponent's habits and playing methods and formulate strategies through historical battle data. If the bureau adopts a new routine against the opponent and the actual performance is very different from the previous data, the strategies formulated based on the previous data will be useless, Lost relevance. Although the current domestic live broadcast software can broadcast the real-time combat situation of the game through video, it lacks the quantitative display of specific data. Only the qualitative live video display. Relying on the subjective qualitative information obtained in the video, it is impossible to make an objective and quantitative description of the situation on the field, so it is impossible to put forward an accurate and quantitative Tactical Guidance Scheme, So as to lose the initiative of the game and hand over the chance of victory to others.

2、 Server communication framework design

The real-time data release software adopts server client structure to manage real-time combat data on the server.

（1） Multi user communication framework based on socket

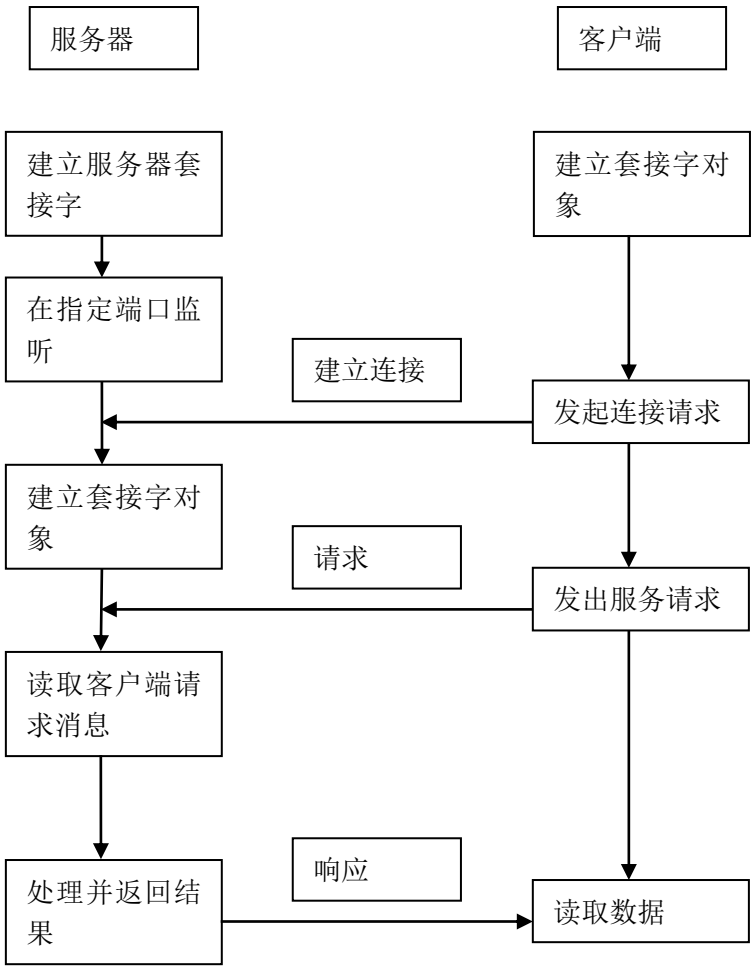
Scheme 1 uses socket based multi-user communication framework to push real-time battle data and connect multiple clients.

After java was created, its multithreading ability, Internet connection ability and strong cross platform ability have amazed everyone in the new era. The strong combination of Java technology and the Internet has made Java and the Internet germinate and grow rapidly and show strong vitality. Under the background of the continuous evolution of Internet technology, the information interaction between systems on different platforms has become more and more important. At this time, multi-user

information transmission, a very noticeable field of Internet behavior, also brings more challenges to the application programs running on the Internet. Java language has strong platform independence. Its network programming function and multithreading mechanism have become the primary consideration for realizing network multi-user communication [1].

1. Socket communication mechanism in lol real-time combat data publishing software

The server side of lol real-time combat data publishing software transmits data with Android client through TCP protocol implemented by socket. A socket in the server is uniquely determined by an IP address and a port number (port 9898 is used in this design). Use socket to establish connection between real-time data server and Android client. The server side of the real-time data publishing software has been listening. When the Android client of the real-time data publishing software sends a connection request to the port monitored by the server (port 9898 is used in this design), the real-time data server program returns a socket object containing the client socket information, which realizes a special virtual connection channel with the Android client. The Android client program adds the request content to the socket. The real-time data server program receives the request content through the corresponding socket, and then parses the data and returns it to the Android client program. When the information interaction between the Android client and the server is completed, close the virtual connection channel. The communication mechanism is shown in Figure 1.



(the server client establishes the server socket, listens on the specified port, establishes the socket object, reads the client request message processing and returns the result, establishes the socket object, initiates the connection request, sends the service request, reads the data, and establishes the connection request response)

Figure 1 native socket communication mechanism

When the lol real-time data publishing Android client connects to the real-time data publishing server through port 9898, the server has a unique socket object that can track and monitor the information interaction with the Android client. The real-time data publisher uses this monitoring and tracking method to communicate with multiple Android clients using port 9898 to transmit data. All socket objects of Android clients connected to the server are owned by the real-time data publisher. The lol real-time data publishing server can distinguish the data request contents from different Android clients. It gives different push contents to the request contents of different Android clients through different socket objects. The lol real-time data publisher uses Java's multithreading mechanism to open up multiple threads, which handle the

network I / O operations of different Android clients, so as to push the concurrent real-time battle information of multiple Android clients.

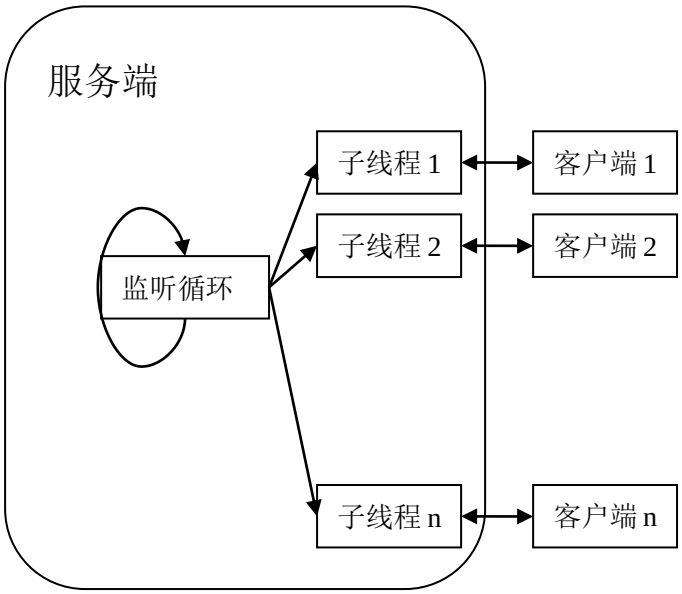
The server and Android client of lol real-time data publishing software adopt the socket communication mode based on Java. The real-time battle data subscription and push code written in this scheme can read the real-time battle data subscribed by Android client from the socket and write real-time battle data to the socket like file I / O, Very easy to use network resources. The underlying mechanism of the server side of lol real-time data publishing software is to loop the port listening in the main function of the real-time data server and use the blocking function to respond to the connection request of the Android client.

2. Processing principle of real-time battle data push server

First, establish a socket bound to port 9898: socketserver socketserver = new socketserver (9898); The real-time battle data server circularly listens to the connection request initiated by the Android client using port 9898: socket = ServerSocket. Accept(); When the Android client initiates a request, the accept() function returns a socket object that stores the Android client information to complete the connection between the Android client and the battle data server. After the battle data server accepts the connection, use the getinputstream() and getoutputstream() methods in the obtained client socket object to establish file I / O data stream objects inputstreamreader and outputstreamwriter to communicate with the Android client, so as to realize the data interaction between the battle data server and the Android client.

3. Processing principle of concurrent connection of Android client for real-time battle data

When an Android client initiates a connection request to the battle data server, the battle server starts the session with the Android client, but the blocking operation will be performed because it needs to wait for the subsequent request of the client. In order to realize the communication between the battle server and multiple Android data publishing clients at the same time, the battle data server uses the multi-threaded processing mechanism of Java, The data server circularly listens to port 9898. Each time it receives the connection request initiated by the client, it creates a socket object and creates a new thread. It hands over the socket object with Android client information to the newly established thread, and uses this specific new thread to complete the communication with the data publishing Android client. The main function of the server continues to execute the loop structure and listen to port 9898 until the main function terminates and the server is shut down. (as shown in Figure 2)



(the server listens to loop sub thread 1, sub thread 2, sub thread n, client 1, client 2, client n)

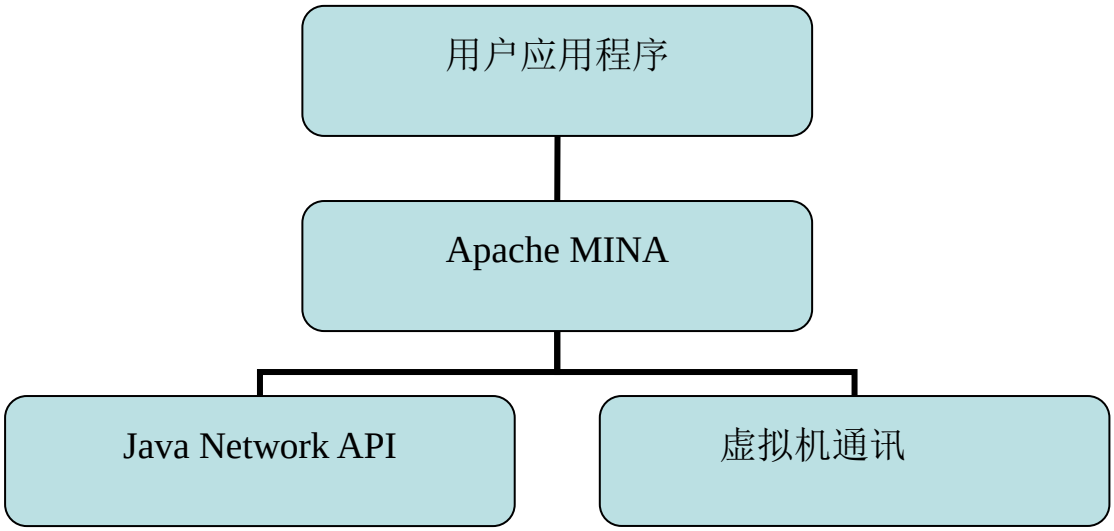
Figure 2 block diagram of implementation mechanism of native socket multi client connection

（2） Using Apache Mina server network communication framework

Scheme 2: use Apache Mina server network communication framework to push real-time battle data and read subscription data sent by Android client

With the increasing number of users using this software to monitor real-time combat data, higher and higher requirements will be put forward for the server-side program of real-time combat data. It will also become common for a server to connect hundreds of thousands of Android clients at the same time. This challenges the high performance of server-side programs. Therefore, in this design, we need to use an efficient network I / O bottom layer to develop high-performance network applications more easily. The emergence of Java NiO and Mina provides a solution for this design. Mina adopts asynchronous I / O and event driven mechanism, which has high efficiency and performance. Combined with Java NiO, this design can use Mina to easily develop high-performance real-time combat data server program.

1. Implementation principle of real-time combat data server using Mina framework



(user application Apache Mina Java Network API virtual machine communication)

Figure 3 Mina architecture

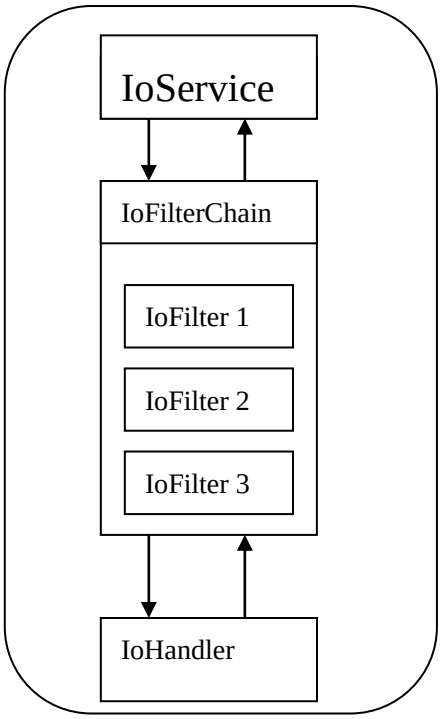


Figure 4 Mina detailed architecture

The real-time battle server program uses the ioservice class in the module chain as the entry of the server application. Code: niosocketacceptor acceptor = new niosocketacceptor();

2. The server encodes and decodes the IO stream

The server program uses I / O filter chain to filter the input and output IO streams. Code: acceptor. Getfilterchain(). Addlast ("codec", newprotocolcodecfilter (New mytextlinefactory()));

The IO stream data is encoded and decoded in a user-defined way, which is implemented through mytextlinefactory.

Create a new class mytextlinefactory, inherit the interface protocolcodecfactory, and rewrite the getdecoder and getencoder methods to correspond to the decoding and encoding methods respectively. The specific implementation method is to establish a class named mytextlineencoder, inherit the interface protocolencoder and realize coding; Establish a class named mytextlinedecoder, inherit the interface protocoldecoder, and realize decoding. Return an object of mytextlinedecoder in getdecoder and an object of mytextlineencode in getencoder.

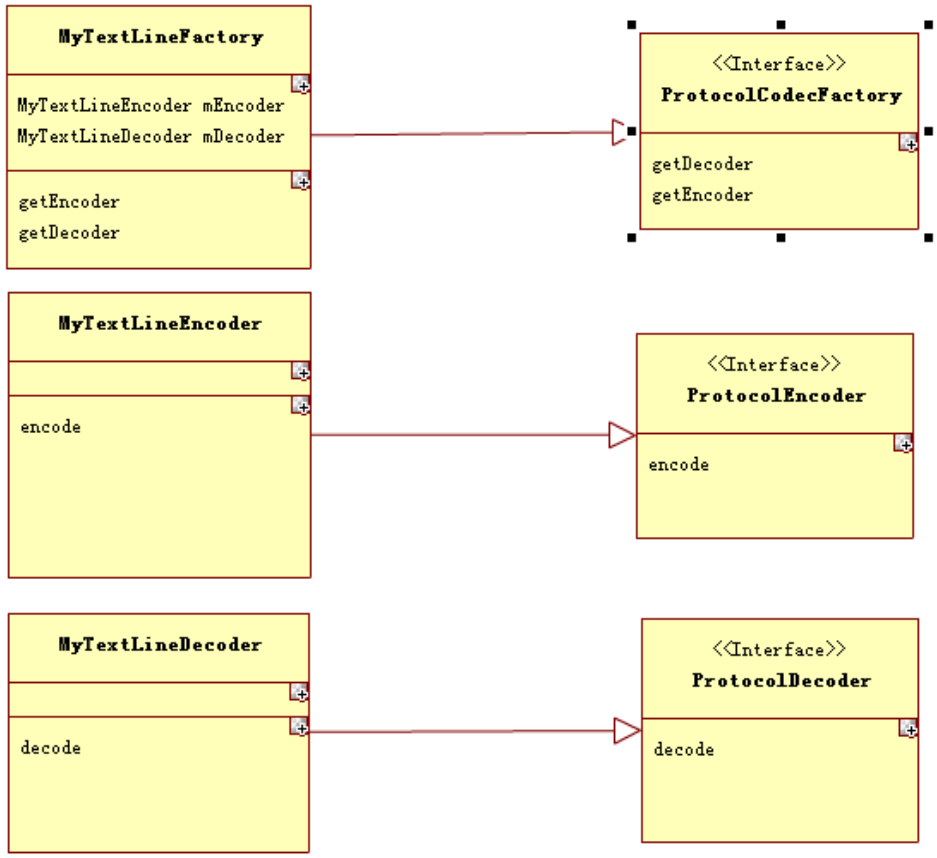


Fig. 5 filter class diagram

(1) Server coding mode

Because the sent content may be an object, but only bytes are allowed to be transmitted at the network layer, the sent message must be encoded into a byte string and rewrite Protoco in mytextlineencoder-

-The lencoder interface function realizes the conversion from message object to byte. The method is as follows: first, judge the object message of object type to be sent. If it is not of string type, convert it to string

type; Then transcode the converted message object, use the allocate method of iobuffer object to open up a section of memory with the length of message.length, use the setautoexpand method of iobuffer to realize the dynamic expansion of memory, and use the putstring method of iobuffer to encode the message converted into character string according to the system's default UTF-8 encoding method. The flip function is used to indicate the end of the conversion, and the write method in an object of protocolencoderoutput type is used to send the transcoded iobuffer object.

(2) Server decoding mode

Since the data transmission in the network layer is in the form of byte stream, rewrite the protocodecoder function in mytextlinedecoder to decode the received data and restore it to a string. The specific decoding method is: first read the received content, and then convert the received bytes into strings. The specific method is as follows: first, use an integer variable startposition to record the position where the byte stream is started to be read, then start a while loop, use the hasremaining method in iobuffer to judge whether there is still data in the byte stream and decide whether to continue the loop. The content of the loop body is to use the get method of iobuffer to read one byte B in the byte stream each time, If the byte we read is \ n, it means that the reading is over, and then the end processing is performed. At the end of processing, first use the position method of iobuffer to record the current read position CurrentPosition, and then use the limit method of iobuffer to record the current total length, and then perform interception to intercept all data between the start position and the end position of a line. The position and limit methods in iobuffer are used for redirection, and the slice method is used for interception. The intercepted iobuffer object is stored in the byte array, and then the byte array is converted to string type. The method is to use the string constructor to write the obtained byte array into the initialization list of the Stirng constructor. Finally, use the write method in an object of type protocoderoutput to send the transcoded iobuffer object. After the interception, use the position and limit methods to restore the position, and restore the position from startposition to CurrentPosition to ensure the correctness of the next interception and avoid falling into a dead loop.

3. Service side business processing module

The real-time data server uses iohandler class as the business processing module. In the business processing class, you don't need to care about the actual communication details, just deal with the information transmitted by the client. In order to simplify the handler class, Mina provides the iohandleradapter class, which only implements

the iohandler interface. Therefore, just rewrite the corresponding method in the specific business logic. Iohandler method is shown in the figure below.

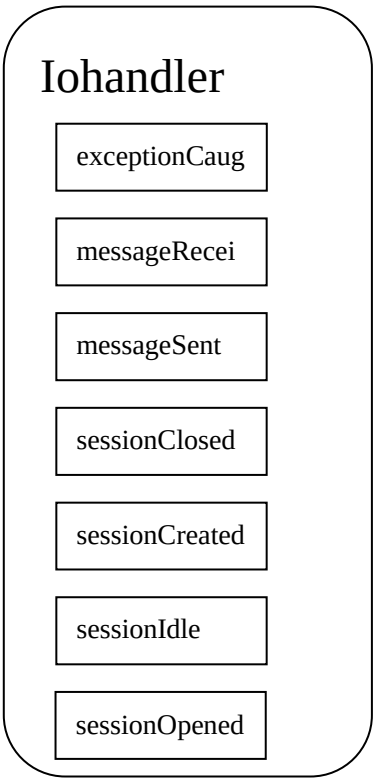


Figure 6 iohandler encapsulation method

Iohandler encapsulates the processing of different events from the client. If you are interested in an event, you can implement the corresponding methods. When the event occurs, the methods in iohandler will be triggered for execution. Iohandler has a total of 7 methods corresponding to 7 events:

（a）void exceptionCaught(IoSession session, Throwable cause)
Triggered when an exception occurs.
（b）void messageReceived(IoSession session, Object message)
Triggered when a message arrives. Message represents the received message.
（c）void messageSent(IoSession session, Object message)
It is triggered when sending a message, that is, it is triggered when calling iosession. Write(). Message represents the message to be sent.
（d）void sessionClosed(IoSession session)
It is triggered when the connection is closed, that is, when the session is terminated.
（e）void sessionCreated(IoSession session)
Triggered when a new connection is created, that is, when a new session is started.

（f）void sessionIdle(IoSession session, IdleStatus status)

Triggered when the connection is idle. Use the setidletime (idlestatus status, int idletime) method in iosessionconfig to set the idle time of the session. If the idle time of the session exceeds the set value, the method is triggered. You can obtain the number of times the sessionidle is triggered through session. Getidlecount (status).

（g）void sessionOpened(IoSession session)

Triggered when a connection is opened. In the current implementation, it seems that there is no great difference between sessionopened and sessioncreated. Sessioncreated is triggered before sessionopened.

Iohandler is an interface. Generally, it is not necessary to directly implement each method of the interface. Mina provides an iohandleradapter class, which implements the methods required by iohandler, but does not do any processing. The real-time battle data server implements its own iohandler by extending iohandleradapterhandler, and rewrites the event method. [2]

By rewriting the messagereceived method in iohandleradapter, the client subscription information is parsed. The specific analysis method is:

The subscription data is divided into three segments. The first segment indicates whether the requested content is currently online, the second segment indicates the type of subscription order, and the third segment indicates the ID number of the specific person requesting.

The corresponding situation of the second paragraph of data is shown in the table below:

Table 1 subscription list code

Subscription content	Subscription order code
Real time economic capability of the enemy and ourselves	0
Real time vision control	1
Real time control of epic neutral creatures	2
Resource control real-time data	3
The enemy and our team contribute to the real-time participation rate from the enemy and our side	4
Output rate	5
Money ratio	6
Sustained injury rate	7
Real time kDa value	8
Replenishment & real time number of neutral creatures	9
Real time number of epic neutral creatures killed	a

Real time hero data of the enemy and ourselves are distributed through real-time death	b
Real time kill distribution	c
Gain money value	d
Real time kDa value	e
Real time troop replenishment value	f
Real time loading	g
Real time use of key skills	h

According to different request contents, the server will package and send different data to the client, so as to push specific data.

According to the comparison of the above two schemes, we find that the multi-user communication framework scheme based on socket is relatively slow in dealing with multi client connections, because it uses blocking I / O for network communication. If 100 clients make concurrent connections, 100 threads must be established for processing, which increases the burden of the server and reduces the communication speed. The Mina framework scheme performs well in handling multi client connections, which is conducive to the processing of high concurrent connections. Moreover, the Mina framework separates each module, improves cohesion, reduces coupling, and is conducive to subsequent secondary development. Therefore, scheme 2 is adopted in this design.

（3） Server data management

1. Establishment of battle data model

The server establishes a data class as a data model to simulate the data generated by real users in wartime. The data generated by users in the real battle is mainly divided into five aspects: team position, team role, killing and death, causing or bearing damage, and others (amount of treatment, money, etc.).

Kill data includes: the number of heroes killed, the number of character deaths, the number of assists, and the maximum number of consecutive kills.

The damage data includes: damage to heroes, physical damage to heroes, and magic damage to heroes.

Taking damage data includes: restoring health, taking total damage and taking physical damage. Take magic damage.

Other data include: the amount of money obtained, the number of defense towers destroyed, the number of small soldiers killed, the number of neutral creatures killed, vision control, the number of dragons killed, and the number of dragons killed.

Team location data is divided into: previous order, middle order, shooter, auxiliary and field play.

Team roles include: tank, physical output core, magic output core and therapeutic auxiliary.

2. Operation mode of battle data simulator

The server uses HashMap to manage the data objects of all users, and uses a timer to change the data of all users once a second.

The method of simulating data by battle data simulator is as follows:

The money of all users is increased by one per second, the number of small soldiers killed is increased by one every ten seconds, the number of neutral creatures killed is increased by one every three minutes, the amount of magic and physical damage is increased by 100 per second, and the number of life restored is increased by three per second.

The team character is the user of the tank who takes a random physical damage of less than 1000 and a random magic damage of less than 1000 per second. Taking total damage is taking physical damage plus magic damage.

The user whose team character is the core of physical output increases a random physical damage to the hero within 500 per second. Users whose characters are the core of magic output will increase magic damage to heroes by a random number of less than 200 per second. Users whose role is therapeutic assistance increase the amount of treatment with a random number of less than 200 per second.

If the team position is the user of the previous order, the number of reinforcements per second will increase by one, and the money will increase by a random number within 20; if the position is the user of the middle order, the number of reinforcements per second will increase by a random number within 30, and the number of reinforcements per second will increase by one; if the position is the user of the shooter, the number of reinforcements per second will increase by a random number within 30, and the number of reinforcements per second will increase by two. For users whose position is playing wild, the money increases by less than 10 random

numbers per second, and the number of neutral creatures killed per second increases by 1.

3、 Android client design

（1） MVC mode of Android client

The client uses the list control listview as the main form of the interface. As a container for displaying the specific users currently online, listview control uses MVC mode to separate the front-end display from the later data. That is, when loading data, the listview control does not directly use the add or similar methods of the listview class to add data, but specifies an adapter object. This object is equivalent to C (controller) in MVC mode. Listview is equivalent to V (view) in MVC mode and is used to display data. The list or array that provides data for the listview is equivalent to the M (model) model in the MVC schema. [3]

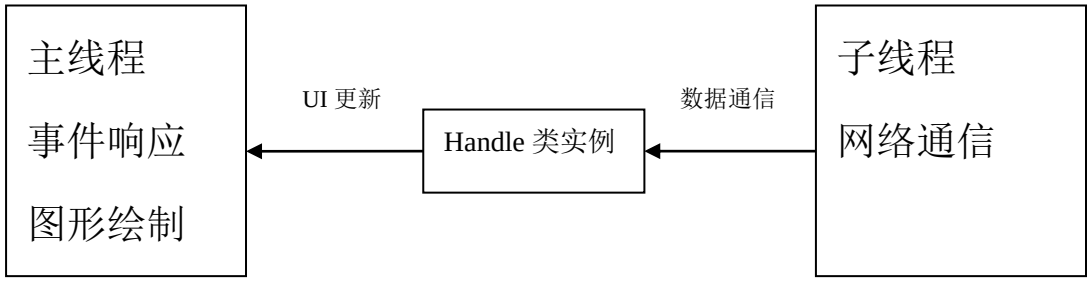
In the listview control, obtain the data to be displayed through the controller (adapter object). When creating an adapter object, you need to specify the data to be displayed (list or array object), so the data to be displayed and listview are connected through the adapter object and independent of each other. That is, the listview only knows that the displayed data comes from the adapter, and does not know whether these numbers come from the list or the array. For data, you only know to add these data to the adapter object, and you don't know that these data will be used for listview controls or other controls.

Android real-time data client uses ArrayList as the data model. On the one hand, it manages the current online user data received, on the other hand, it manages different types of user subscriptions, adds them to the controller adapter object, connects with listview and displays them on the interface.

（2） Network communication module of Android client and inter thread communication

1. Inter thread communication

The real-time data publishing client uses the listview control in the main thread (i.e. UI thread) management interface to distribute events and complete the graphics drawing of real-time data. The sub thread is used for network communication with the server, and the handler is used to realize the communication between the sub thread and the main thread, so as to update the data.



(main thread event response graph drawing sub thread network communication h andle class instance data communication UI update)

Fig. 7 Schematic diagram of communication between sub thread and main thread

When the client application starts, Android will first start a main thread (i.e. UI thread). The main thread is the UI control in the management interface to distribute events. For example, click an item in listview in the client, and Android will distribute events to the item in response to your click operation. If a time-consuming operation is required at this time, for example, when reading data online or reading a large local file, these operations cannot be placed in the main thread. If the network I / O operation is placed in the main thread, the interface will appear suspended. If it has not been completed for 5 seconds, an error prompt "forced shutdown" will be received from the Android system ". at this time, the client will flash back. At this time, we need to put these time-consuming operations in a child thread, because the child thread carries out network communication and receives data, which involves UI update, while the Android main thread is thread unsafe, that is, the UI can only be updated in the main thread, and the operation in the child thread is dangerous. Therefore, the real-time data publishing client The handler is used to solve this complex problem. The handler running in the main thread (UI thread) and the child thread can transfer data through the message object. At this time, the handler is responsible for receiving the message object (which contains data) transmitted by the child thread (the child thread uses the sedmessage method) , put these messages into the main thread message queue to update the UI with the main thread.

2. Network communication module

Use the native socket class for client network communication. The specific methods are as follows:

Declare a private socket type member variable in the activity class, and start another thread to initialize it. Use socket (InetAddress) address, int Port) method to create a stream socket and connect it to the specified port number of the specified IP address. For the Android simulator on the PC side, the IP address 10.0.0.2 is used, and for the Android Software on the mobile side, the real IP address of the host is used, both of which are connected to port 9898.

Client network communication coding process: use socket. Getoutputstream() method to get the output stream of this socket, and use this output stream to create the default character coding (UTF-8) Outputstreamwriter of. Outputstreamwriter is a bridge between character flow and byte stream: the characters to be written to the stream can be encoded into bytes using the specified charset. In this design, the default character set is directly used to match the server. In order to obtain the highest efficiency, the client wraps the obtained outputstreamwriter into bufferedwriter and uses bufferedwriter (writer) out) Method to create a buffered character output stream using the output buffer of the default size. The text received by the sub thread through network communication is written into the character output stream to buffer each character, so as to provide efficient writing of single character, array and string, so as to avoid frequent calls to the converter. Finally, a printwriter is created according to the existing bufferedwriter to the server Send a subscription.

Client network communication decoding process: use the socket. Getoutputstream() method to get the input stream of this socket, and use this input stream to create an inputstreamreader using the default character set. Inputstreamreader is a bridge for bytes to flow to the character stream. It reads bytes and decodes them according to the default decoding method (UTF-8) Decode into characters. In order to achieve the highest efficiency, the client wraps inputstreamreader in BufferedReader, reads text from the character input stream of sub thread network communication, and buffers each character, so as to realize efficient reading of characters, arrays and lines. Use the readLine method in BufferedReader to read a message sent by the server.

After entering the real-time graphics rendering interface, the sub thread will cycle to read the data sent from the server. The message header sent by the server contains the ID number of the sent data, which is used to indicate which user's real-time battle data is sent. The client uses a boolean variable to control whether to prohibit or allow the reception of data. When exiting the graphics rendering activity, the o of the activity Set the variable in the ndestroy method to prohibit accepting

data in the background. In addition, the shutdown of the server will also cause the client to stop accepting data.

Before performing network I / O operations, the sub thread creates an instance of the message class. After receiving the server packet, it stores the ID number in the integer variable arg0 of the message, extracts the specific data and stores it in the object class object obj of the message, and then sends the message to the main thread using the handler.sendmessage method. After receiving the message, the main thread executes the updatechart method, which is more convenient The new chart realizes the real-time drawing of graphics and displays the real-time data of the server in the form of chart.

（3） Client graphics rendering

1. Introduction to drawing method

The real-time data publishing client uses three ways: real-time bar chart, pie chart and line chart to display the enemy's strength, team contribution and hero data.

The Android chart engine achartengine is used to draw different kinds of dynamic charts. Taking the drawing of broken line chart as an example, the code idea is briefly introduced:

Firstly, initialize the chart in the oncreate method of activity, and use the chartfactory factory class to build different types of chart container controls to render different types of chart objects.

Create three new methods in the activity class, getdataset(), getrenderer(), and updatechart(), which are respectively used to obtain the dataset container, renderer container, set chart parameters, and cooperate with sub threads to update the chart.

The implementation of getdataset is as follows: first, create a xymultipleseriesdataset type dataset container to store the data sets of multiple curves, then create several xyseries type datasets to store the curve data, initialize the name of each curve, and finally add the datasets of all curves to the dataset container.

The implementation method of getrenderer is to first establish an xymultipleseriesrenderer type renderer container to initialize the coordinate system, grid, title, etc., and also to store the renderer of multiple curves. Then create several xyseriesrenderer type renderers to store the parameters of curves, such as line color, stroke point size,

linetype, etc., and initialize each The name of each curve, and finally add the renderer of all curves to the renderer container.

Updatechart is implemented by using the removeseries method to remove the old point set in the dataset, and then adding the new point set obtained from the server in the dataset. After putting the new point set into the dataset, use the invalidate function to update the view.

2. Introduction to client presentation

The design client mainly displays the battle data in three aspects: the strength comparison between ourselves and the enemy, the team contribution rate of both sides, and the hero data of both sides.

Among them, the strength of the enemy and ourselves is displayed in three dimensions from four aspects: the real-time economic ability of the enemy and ourselves, the real-time vision control, the real-time control of epic neutral creatures and the real-time data of resource control. Users can formulate the next tactics based on this objective data. Take an actual game as an example. After 30 minutes of the game, the user inquired in the lol real-time battle data release software. Now our economic strength is not as good as the enemy, but our field of view control ability is much better than the opposite side. In the next step, we can focus on the use of ambush tactics, preempt the enemy's field of view in the unknown area, and destroy the enemy through ambush, So we can win the game. Some display charts are as follows:

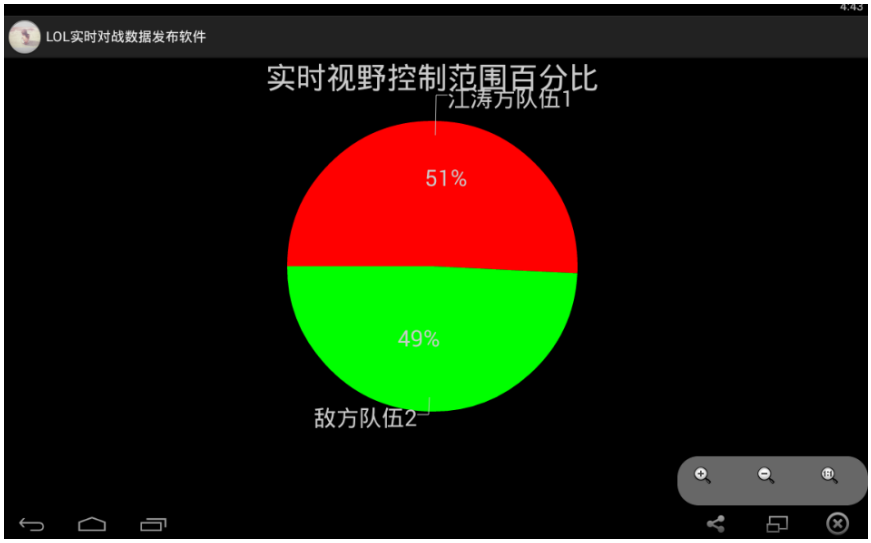


Figure 8 real time visual field control

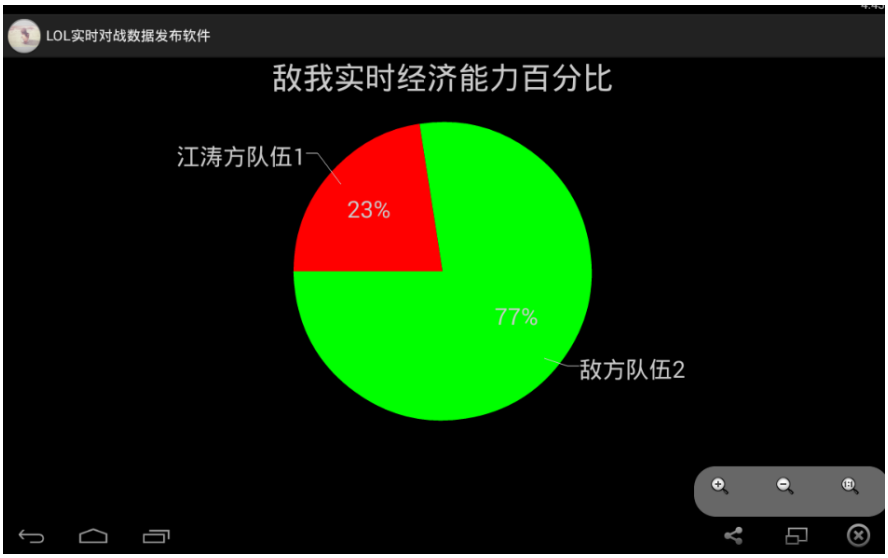


Figure 9 Comparison of real-time economic capability

The contribution of the enemy and our team comprehensively shows the real-time contribution of each person to the whole team at the current time from the aspects of the enemy and our real-time team participation rate, output rate, money ratio, damage bearing rate, real-time kDa value, the real-time number of replenishment & neutral creatures, and the real-time number of epic neutral creatures, so as to provide a clear and optimal idea for group warfare and resource grabbing. Taking a qualifying game as an example, when the game lasts for 40 minutes, the outcome of the group battle in one game can be determined at once to determine the victory of the whole game. At this time, the user observed in the lol real-time battle data release software that the middle singles and wild heroes opposite now have the highest contribution rate of the team and the highest output damage in the whole game, In the next step, we can give priority to killing the two heroes and make targeted defense equipment. In the next group war, we can specify an accurate killing scheme and successfully solve the enemy's core soul figures first, so as to win the group war and lay the foundation for the victory of the game. The actual interface is shown as follows:

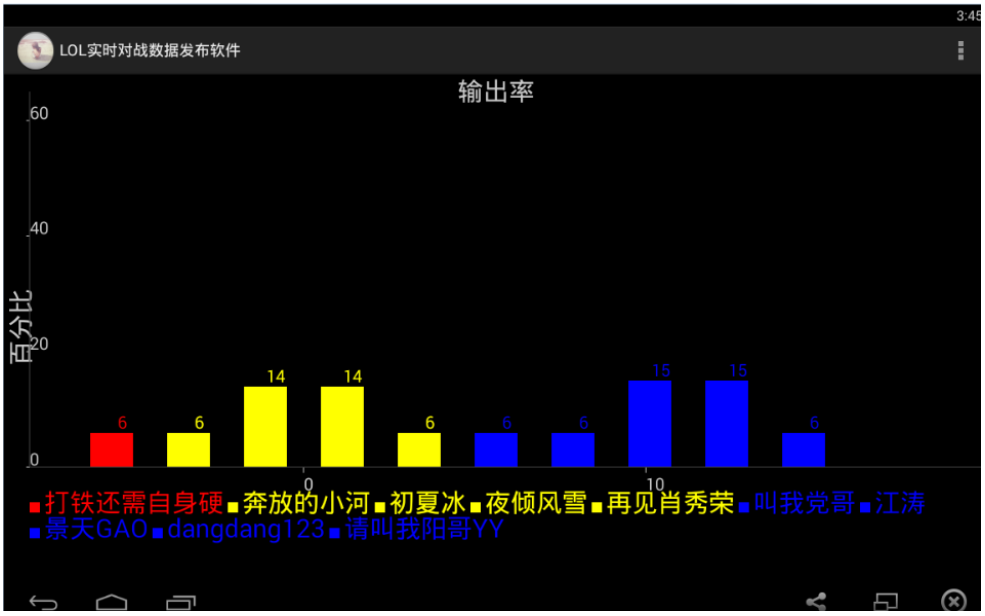


Figure 10 output rate

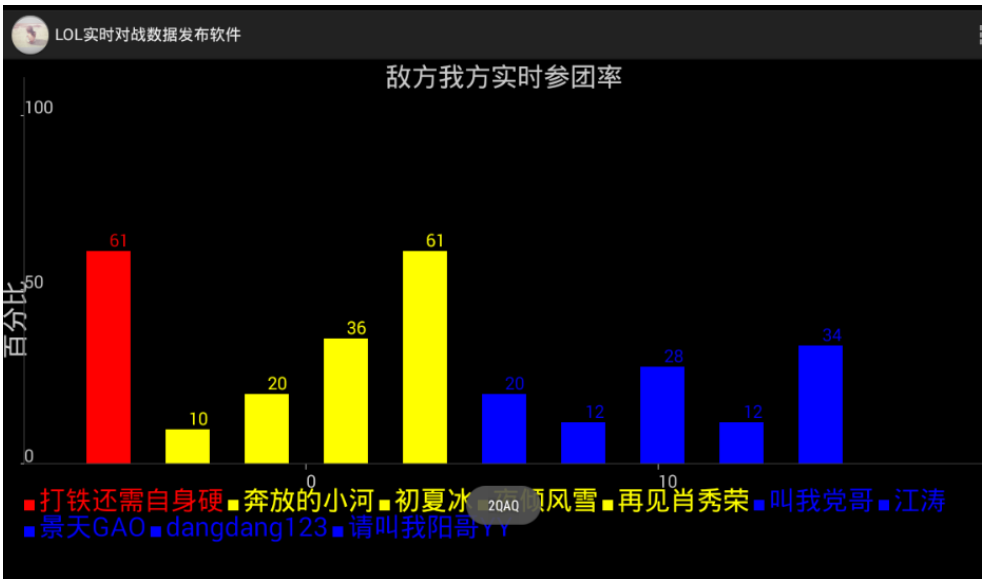


Figure 11 attendance rate

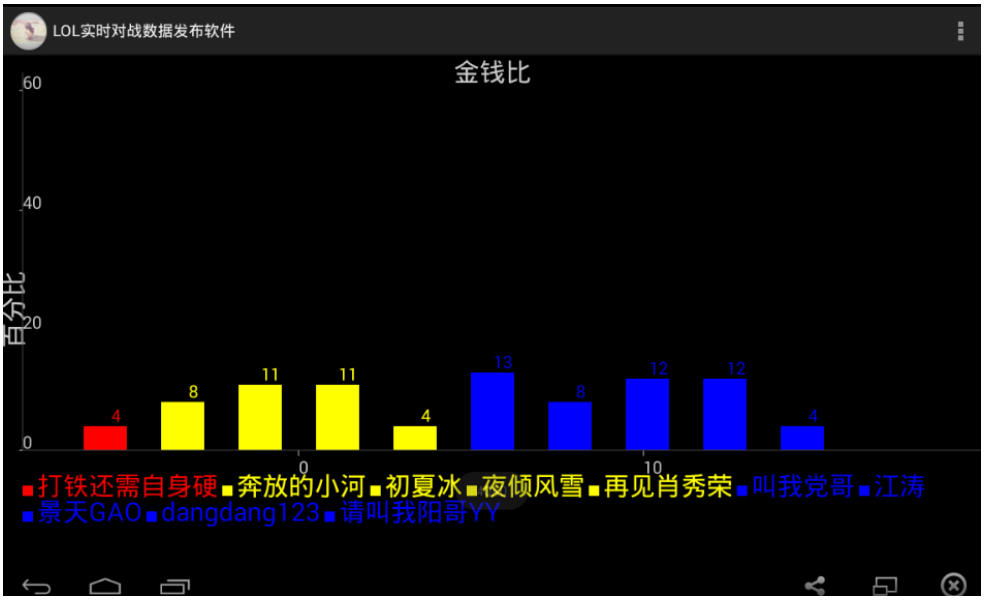


Figure 12 money distribution

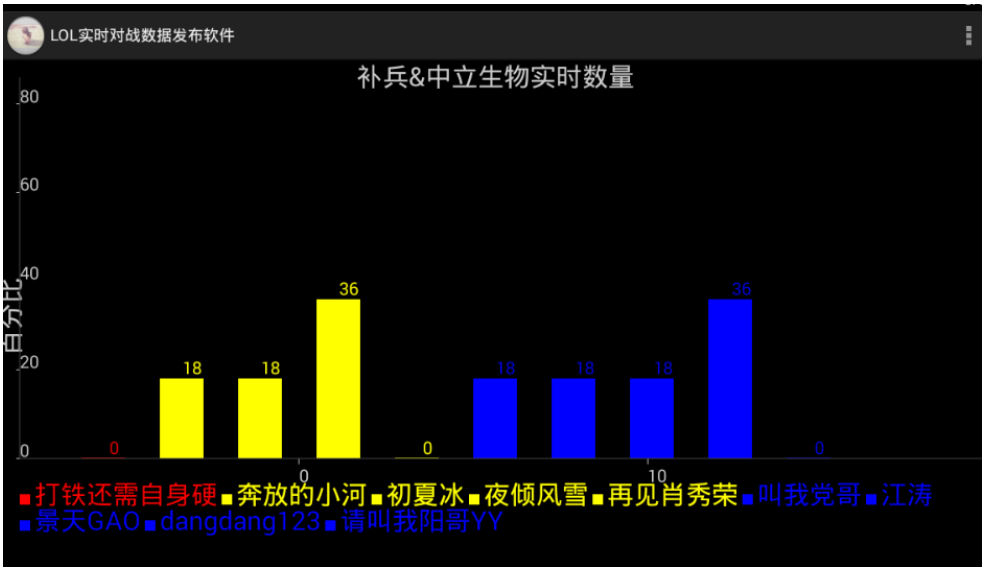


Figure 13 number of creatures killed

The real-time hero data of the enemy and ourselves are displayed through real-time death distribution, real-time kill distribution, obtaining money value, real-time kDa value, real-time replenishment value, real-time loading, real-time use of key skills and so on. By viewing the real-time hero data of this game, we can formulate effective targeting strategies according to the characteristics of enemy heroes, so as to limit the play of the core figures of the enemy team, reduce the team contribution rate of the core figures of the enemy team and improve the winning rate of this game. Take a game as an example. When the game lasts for 10 minutes, the game has just begun. You need to find a breakthrough to quickly open the situation and establish an advantage. At this time,

the user observed in the lol real-time battle data release software that the two key life-saving skills of the opposite legal system output hero are cooling, In the next step, we can focus on specifying a targeted killing scheme for the hero, which successfully opened the deadlock and established a good starting advantage for the game. Some interfaces are shown as follows:

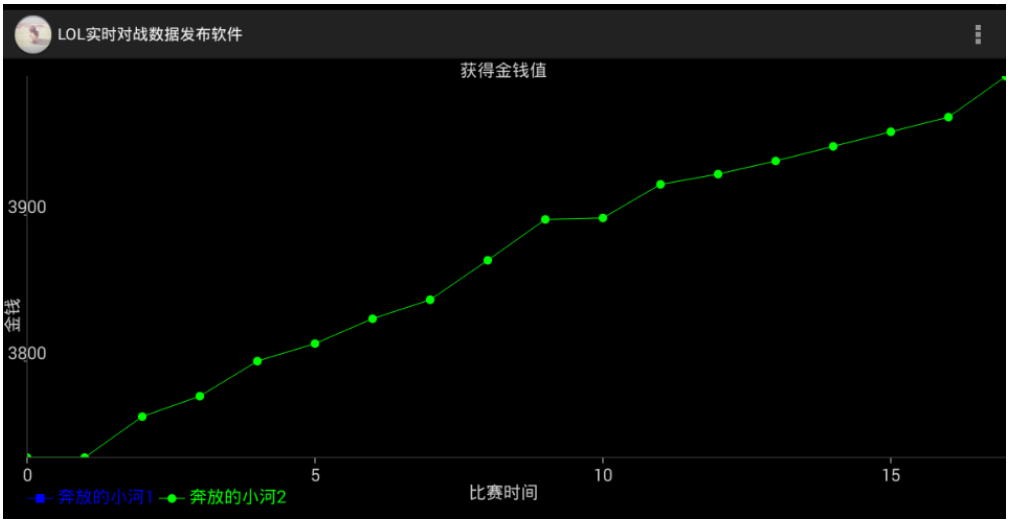


Figure 14 getting money value in real time

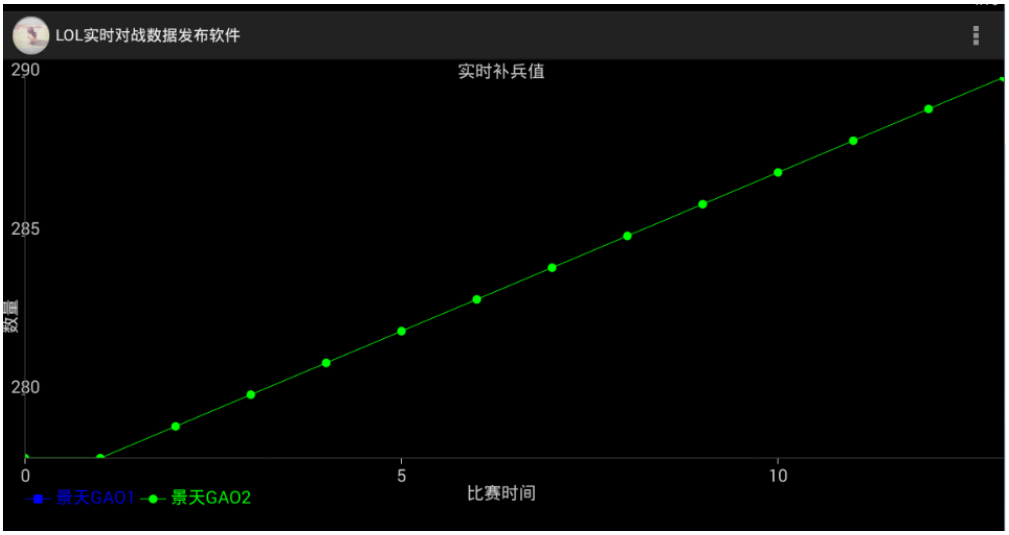


Figure 15 number of real-time reinforcements

3. Software testing

The client can successfully obtain the online user name of the server, and there is no bug in the jump between activities. The actual client operation effect diagram is as follows:

Figure 16 client interface test

The server receives subscription data and pushes real-time battle data. The operation interface is as follows:

Figure 17 test of sending and receiving data at the server

4、 Summary and Prospect

This design takes real-time data release as the core function, effectively makes up for the defects of similar domestic software, solves the problem that previous software can not release real-time battle information, and provides users with quantitative data of this battle to the greatest extent within the timeliness of information, so as to help users optimize their next strategy.

According to the user's demand for lol real-time battle data, this design actively delivers the team strength comparison, resource allocation, team real-time contribution rate and other information required by the user to the user. The basic workflow is:

First, the server pushes the current online player and index number to the client. After the client obtains and parses the information, it informs the user of the online player directory information that can be read.

Then, the user selects the real-time battle data content to be subscribed, including the specified viewing players, the required specific battle data categories and other related content, and submits it to the server.

Finally, the server collects relevant battle data according to the user's subscription form, which is packaged and pushed by the server to the client. After the client obtains and parses the information, it informs the user that the information can be read. Its contents include two categories:

The first type is to send only the directory or index notification of the battle data to the user, and the user can further query the corresponding battle data information according to the battle data notification.

The second type is to directly package and send the data itself in the required battle data to the user.

Although after careful design, this paper has basically achieved the expected objectives and achieved certain results, there are still many places that can be improved and new functions in the future.

Firstly, the possibility of software theft can be effectively prevented by joining the secure login authentication process. It ensures that data and files can only be used and viewed by users with permissions,

so that users with different levels of permissions can view different files and data contents.

It can also add the function of simultaneous monitoring of multiple games to realize real-time monitoring and analysis of data of multiple teams.

Finally, we can consider the web-based development of cross platform HTML5, which can make the software run easily in different platforms and reduce the cross platform development cost and cycle.

reference

[1] Tong Ming, Li Zhishu. Multi user communication framework and Implementation Based on socket. Journal of Sichuan University, 2006, 43 (3): 703-705

[2] Yang Tiejun, Xu Hefei, Huang Lin. design of a data communication platform based on Mina framework. Microcomputer information, 2009, 25 (11-3): 22-24

[3] Li Ning. Complete handout on Android development. Beijing: China water resources and Hydropower Press, April 2012. 160 ~ 163

Thank you

Time is always short. In the twinkling of an eye, the four-year university time will end in a hurry. I have received many kinds of help from schools, teachers and classmates, so that I am grateful every time I read it.

On the occasion of the completion of this thesis, I would like to thank Mr. Liang Chenghui sincerely. He gave too much careful guidance in my graduation design stage. It was in the reminders, inspections and discussions again and again that I made progress bit by bit. His rigorous scholarship and selfless work attitude have deeply affected me and become the driving force for my progress.

Thanks to chuyang, he Yuqian, Lang Fudong, Dang Xiangwei and other students. It is precisely because of your company and support that I can come up with this idea, complete the graduation project little by little, and overcome one difficulty after another.

Thank my dearest parents for your painstaking upbringing and selfless love. It is you who have always supported me, encouraged me and tolerated me. It is you who always help me through difficulties when I encounter all kinds of difficulties. It is difficult for me to repay my life.

Thanks to the school for giving me a learning environment, I spent four years in college, learned new knowledge and had a broader vision.

Gao chuyang

2016 in Weihai,