Research and application of instant messaging APP based on Openfire+XMPP

Hou Yan

School of Computer Science and Technology, Zhoukou Normal University, Zhoukou 466001, China;

houyan@zknu.edu.cn

Abstract

In order to solve the problem that communication software based on different communication protocols cannot communicate with each other, XMPP based on XML protocol appeared. This project adopts open source Openfire as the service, and develops the instant communication client based on the Android system based on XMPP communication protocol, which solves the problem that the current heterogeneous instant communication application software cannot communicate with each other, pays more attention to the message security of instant communication and serves the public efficiently. Distributed structure and modular system architecture are adopted to enhance the maintainability and extensibility of the system. Through the implementation of the open source protocol, the system architecture suitable for China's national conditions is constructed and applied in practice.

Keywords

XMPP; Asmack. Android.

1. Introduction

The new smartphone revolution is changing people's lives, and the demand for instant messaging based on Android system is also increasing. With the rapid development of instant messaging, the pursuit of a unified communication format, efficient, cheap and rich and colorful communication tools. XMPP as a successful open source research organization instant messaging protocol, it follows the standard, xml-based protocol, and open, using safe and reliable SASL and TLS technology, using a distributed structure and modular system architecture. Therefore, XMPP protocol plays an irreplaceable role in the development of enterprise, school and other internal communication systems. Developing a fully functional, responsive, xmpp-based instant messaging system that supports service clustering is a consistent requirement for enterprise and school communication software.

2. Research background and significance

2.1 Research background

Instant Message (IM) can send and receive Internet messages in real time. Many communication systems use different communication protocols and cannot directly communicate with each other, while the cross-platform XMPP protocol can solve this problem. XMPP has the advantages of easy extension, distributed structure and modular system architecture, as well as the advantages of low cost, high efficiency and powerful instant messaging. In addition, instant messaging has introduced the diverse functions of file, voice and video into the original single voice. Instant messaging has a strong vitality, for the development of enterprise communication and office to provide excellent information exchange platform.

2.2 Research significance

With the development of science and technology, the popularization of Internet network and the innovation and progress of its corresponding technology, the topic of information transmission between the Internet still becomes the most concerned and thorny issue nowadays. Say goodbye to the traditional telephone expensive way of communication, instant messaging with low cost, efficient, functional diversification deeply attracted everyone, improve work efficiency, bring more economic value. Developers specify a variety of communication protocols, is bound to affect the popularity of communication software and the inconvenience of people's experience, the emergence of XMPP makes the communication protocol has a unified and harmonious standard, will impact the monopoly of individual communication software, usher in a convenient communication situation.

3. Introduction to development technology

3.1 Architecture of C/S pattern

C/S (Client/Server, Client/Server), the Client mainly provides a friendly interface to the user to display the data received from the service. The user can pass the relevant request to the Server through the Client and receive the response from the Server at the same time. The server is mainly responsible for receiving and processing client requests, and responsible for the persistence of data. This system is based on the message modular structure of C/S design, its structure is clear and simple.

3.2 XMPP protocol basis

XMPP agreement

A standard was developed based on the XML protocol to address the chaos of instant messaging, formerly known as Jabber, an open-source form of organization that produces network instant messaging protocols. Used for instant messaging and online live message detection. An important reason why clients can communicate on different servers is that XMPP can build server integration capabilities. Message data passed between clients and servers of XMPP must not only be forwarded from XMPP server, but also support DNS routing between servers.

XMPP protocol cluster

XMPP protocol cluster is XMPP, SASL, TLS and TCP from top to bottom. Between XMPP and TCP, there is Simple Authentication and Security Layer, which provides the mechanism of Authentication, data integrity check and encryption. In order to ensure the Security of XMPP information and data transmission, TLS Transport Layer Security is introduced.

Address space

JID(Jabber Identifier) is the entity used to describe XMPP, whose data grid: JID =[node" @"]domain["/"resource], including domain name, node node name and resource name. Such as cyber@cyberobject.com/res. Domain represents the domain name, which is the primary and necessary element in JID. That is to say, even if there are no other elements, only one domain name JID is legal, which is used to represent the gateway and master server in the network. Node, an optional element, is usually used to represent the client element, which in this case is the user name, and the entity it represents must depend on the domain name to be meaningful. Resource, which is also an optional element, represents the location or device that belongs to the user.

3.3 Openfire

Openfire is a powerful instant messaging and chat server written in the Java language, implements the XMPP protocol, and is open source. Openfire installation and use are easy to use, through the form of Web management, server kernel mainly includes connection management module, the server connection management components, session management, login, registration management components, management, updating, external components, data storage management, file transfer components and transmission components and other components. Any IM client that supports XMPP can connect to the Openfire server.

3.4 Asmack

Asmack is an open source, easy-to-use XMPP (jabber) client class library written in the JAVA language. Asmack provided a basic implementation of the client and provided a good plug-in architecture so that new functionality could be developed in the form of plug-ins without affecting the existing architecture. Sometimes, though, you have to customize the content of the XML file you send to make it work for you. Asmack is a portable version of Smack on Android, suitable for Android development environment.

4. System analysis and design

4.1 System structure

XMPP protocol adopts C/S, namely client/server architecture. Figure 1 summarizes three communication methods: client and server, server and server, client and non-xmpp protocol client. Client and server: the client sends a message request to the server, and after the server responds, it will check whether the client is in the server. If it is, it will start the session. Otherwise, it will respond with an error message. Server and server: when a server requests a session from another server, the domain name resolver parses the other server that is responding to the same request. Client and non-xmpp client: at this stage, the role of the protocol gateway is to coordinate the protocol directly. When XMPP client sends message requests such as session to non-xmpp client, it is through this protocol gateway to coordinate. The purpose is to exchange information between different clients.

4.2 System design

Public module design

XML processing module design: corresponding xml-based XMPP protocol has a fixed format, so the information in XML nodes is stored in the form of JavaBean, and the XML language is parsed by PULL parser technology.

Encryption module design: for XML nodes on the information is easy to parse, but to protect Important information, so the introduction of encryption technology, the system MD5 encryption information on the relevant nodes, the first node information through binary code into BASE64, and then through encryption into MD5.

Functional module design

This module mainly includes registration, login, friend information display, conversation and other functions. Registration: it is necessary for the user to enter the user name, password and other information before using the system. After submitting the corresponding request information, the server should respond to the correct return code and insert the user name and other information into the database. Login: the login process verifies the user's identity information. The server will analyze the user's information and match it to the database. If the response is consistent, the user can enter the system. Friend display: through the display of the status of friends, online, leave or busy feedback, and friends in the group can be randomly assigned, new friend application, friend delete, friends related information display. Session communication: the information session communication between friends. When establishing a connection, judge whether the other party's JID is in the local region before the message can be connected. Otherwise, it shall be forwarded to the destination server immediately. It synchronizes with the server in real time, even if the connection is broken, there will be a corresponding reconnection mechanism.

Database analysis

The system server USES MySQL to store the data exchanged between each other and the data of the system itself. The client mainly USES SQlite to store the data. The tables used in the design of this system are user information table, message table and friend data table. User registry: it is used to record the registration information submitted by users to the server.

5. System implementation

In the development of client function code implementation, to ensure the existence and connection state of ConnectionConfiguration, that is, through the ConnectionConfiguration(String ipAddress,int port) parameters to ensure the connection to the server Ip address (here is the Ip address of the computer) and port (the default is 5222), Or dynamically set its property according to the ConnectionConfiguration, and finally connect to the server through the connetion() method.

5.1 Registered

First let the user enter the user name, if the parameter is empty, prompted "please enter the user", then fill in the password in the same way as here, prompted "please enter the password", in order to ensure the user security password MD5 encryption. After submitting these two data, we have to respond to whether the server registered successfully (the server will query the database whether there is this user name), if successful, it will jump to the login page, and insert the data into the database, otherwise it will give a friendly error reminder, stay on the registration page. First, get an instance of AccountManager, which can be used to create users, change passwords, get user information, and so on, and then pass in the user name and password to register a new user with the server via the createAccount method.

5.2 The login

After submitting the data, wait for the response from the server. If the response enters the main page of the software successfully, otherwise a friendly error reminder will be given. Openfire supports multiple terminal logins. First instantiate XMPPConnection and call its login() to login to the server. Presence.Type has two types: available(online), unavailable(not online), and finally, send status packet to the server through sendPacket() method.

5.3 Message exchange

This part mainly carries on the instant messaging of the message, besides may send the text message, may also send the picture, the speech, the file content, the incoming message will have the order to appear on the external session interface, the newest message will place on the top list. When clicking on a particular user, create a session, first to get the user's JID, determine whether the user in the local server, if not forwarded to the client's server via the Internet, otherwise, to determine the user online status, if online is sending a message, otherwise the message is stored, when the user online, to forward information. Each session is managed by the session management management component, which internally is a session for a thread, and when the server starts, a certain number of threads are assigned to the thread pool. During the session, by sending the Message packet, the to attribute contains the target JID of the Message to be delivered, the body tag contains the session content, and the getBody() and setBody() get the Message content and set the Message content. Chat.sendmessage (String message) creates a message object, which can of course be set to the message object being sent, and calls the Chat class's createMessage() method and sendMessage(message mes). To get the data sent by the other party, in addition to chat.nextmessage (), which can be used to receive messages from friends, you can also set up a message listener via ChatManager.

5.4 Add friends

Add is a function of the system is very important, must first obtain the user's JID and sent to the server, the server to the client first local list to check whether there is a corresponding JID, if any, given this user already exists, or to the server database to query the JID exists, if it does not, prompt "the user does not exist", otherwise can add logic operation. When the system listens to the added user entering the system, a dialog pops up to confirm the operation and ask whether the user is willing to add the friend. On the code implementation, through the Roster createEntry send friend request () method, the parameters corresponding to his best friend's JID, his best friend's nickname, the friend's group (the default is null, it indicates that the wrong friends group limited).

5.5 Roster management

When the user gets the correct response from the server through login, it jumps to the main page. In this system, the first one to enter is the list of friends messages. The next one is the list of friends. Manual grouping, manual deletion, and so on. Friends through the Roster to record information data, data add, delete, search can be through the API to implement such, can through the XMPPConnection. GetRoster () to obtain the Roster instance, Roster display information such as: tiger@team.com or tiger:tiger@team.com [TeamName], the former representative for the group of friends, the latter representatives have group of friends, including TeamName grouping nodes represent the name of the tiger on behalf of the user name. The client to send get packet types of IQ type, to make the server list to return back to the roster. RosterLoaded (Roster Roster) method can be to convey the specified Roster Roster object inside.

5.6 Disconnect and reconnect

When a user encounters a network obstacle while chatting, the system can immediately feedback the corresponding information and save the information sent by the user when the network is disconnected. When the network is normal, the information can be sent to the client to be received. When the network is disconnected, the client will start a thread to listen to the network in real time and inform the state of network information in the form of broadcast. The delay response can be basically ignored. Thanks to the powerful API of Asmack, many functional details have been implemented internally, such as changing the password or logging out. Just get an instance of the AccountManger, pass in the password parameters, and the changePassword(PWD) method can change the password. DeleteAccount () makes it easy to log out of user information. So what we should be looking for is a system architecture, a robust code optimization problem, a good API for a good system.

6. System test

This system is designed based on C/S architecture. It tests the client side and the server side respectively. Modular test is adopted to write corresponding test cases, such as registration module. For the login module, in addition to ensuring consistency with the process, it also ensures that multiple users cannot log in at the same time. Its meaning is to ensure that the system functions and implementation of the same, will not cause unnecessary system collapse.

References

[1] Zhang yan, xia qingguo. Research and application of Jabber/XMPP technology [J]. Science, technology and engineering,2015(06):12-20.10

[2] Dreamtech software development group. Analysis of instant messaging system programming source code [M]. Beijing electronic industry press,2017(03):35-36.

[3] Miao kai. Analysis of security mechanism of XMPP [C]. Communications technology, 2016(08):45-50.

[4] Chen pinghua, li wenliang. Analysis of Android kernel [J]. Modern computer (professional edition),2013(03):23-45.

[5] feng yajun, song zilin, instant messaging system based on XMPP protocol [J]. Military communications technology,2012(12):57-59.

[6] liu zhijun, li xiaofeng. Real-time information mechanism based on SIP protocol [J]. Beijing post and telecommunications daily,2012(03):137-142.

[7] Jim Whitehead,Streaming XML with Jabber/XMPP[Z].Published by the IEEE Computer Society,2011(06):12-65.

[8] Banga G,Druschel p. Measuring the Capacity of a Web Server[Z].USENIX Symposium on Internet Technologies and Systems,1997(02):112-454.

[9] huang yong. Instant messaging moving forward in difference and integration [J], market research, 2013(11):57-99.

[10] Qu chaoyang, lu meilian. Protocol model and application prospect of XMPP [J]. Modern telecommunication technology,2012(3):26-31.

[11] lu zhe, wang shuming. Research on development of enterprise instant messaging platform based on P2P technology [J]. Computer and modernization,2015(06):22-67.