

上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

课程论文

COURSE PAPER



论文题目： 基于 Qt 的聊天程序

学生姓名： 郭进尧

学生学号： 519030910124

课程名称： 计算机通信网络

指导教师： 蒋兴浩

目 录

| | | |
|-------|---------|----|
| 第一章 | 项目概述 | 1 |
| 1.1 | 项目环境 | 1 |
| 1.1.1 | 开发环境 | 1 |
| 1.1.2 | 测试环境 | 1 |
| 1.2 | 功能介绍 | 1 |
| 第二章 | 算法实现 | 2 |
| 2.1 | 服务端 | 2 |
| 2.1.1 | 主要功能介绍 | 2 |
| 2.1.2 | 功能实现 | 2 |
| 2.2 | 客户端 | 3 |
| 2.2.1 | 主要功能介绍 | 3 |
| 2.2.2 | 功能实现 | 3 |
| 第三章 | 测试截图及说明 | 6 |
| 3.1 | 服务器部分 | 6 |
| 3.2 | 客户端部分 | 6 |
| 第四章 | 项目总结 | 15 |
| 附录 A | 部分源代码 | 16 |
| A.1 | 客户端 | 16 |
| A.1.1 | 主页面 | 16 |
| A.1.2 | 找回密码 | 27 |
| A.1.3 | 登录 | 28 |
| A.1.4 | 注册 | 28 |
| A.1.5 | 用户类 | 29 |
| A.2 | 服务端 | 29 |
| A.2.1 | 主页面 | 29 |
| A.2.2 | 用户类 | 34 |
| A.2.3 | 登录 | 34 |

第一章 项目概述

本项目基于 Qt 实现即时通讯聊天程序，包括客户端和服务端，使用的传输协议包括 TCP 和 UDP，通讯方法包括用户间直接通讯（不经过服务器）和经服务器转发通讯。

1.1 项目环境

1.1.1 开发环境

- 系统：Windows10
- 开发平台：Qt creator5.9.2

本项目以 Qt 为开发环境，设计时利用 Qt 独特的信号和槽函数机制以及自带的 UI 界面。用户通过按键发出信号，调用相应槽函数实现功能，使得各个功能函数间相对独立。

1.1.2 测试环境

- 服务端：Windows10
- 客户端：Windows10

1.2 功能介绍

- 用户注册、登录和找回密码功能
- 用户点对点聊天功能
- 多组用户聊天功能
- 用户间文件传输功能
- 用户发送表情功能
- 离线消息保存功能

其中，用户点对点聊天和表情发送基于用户两点间直接建立 socket 传输，不经过服务器转发；而离线消息则由服务器保存，待用户上线后由服务器发送给用户。文件传输功能基于 UDP 协议，通过应用层实现流量控制。

用户注册、登录、找回密码等基础功能基于客户端与服务器建立 socket 传输，服务器保存用户账号信息。

第二章 算法实现

2.1 服务端

2.1.1 主要功能介绍

1. 保存用户账号信息
2. 用户连接
3. 用户注册
4. 找回密码
5. 获取 IP
6. 保存离线消息
7. 获取用户列表

2.1.2 功能实现

在以上功能中，服务器主要扮演聊天的支持者角色，保存用户信息，提供登录、注册、找回密码等账号服务，返回用户 IP 以使得用户可以建立点对点连接。用户与服务器间的信息传递都是通过 TCP 套接字完成。由于一些基础功能涉及的连接通信不多，在此，选择部分功能介绍。

保存用户账号信息：

服务器通过链表结构保存用户信息，并使用文本文档离线存储。服务器启动时，读取存储信息，更新用户链表，表中的每个结点分别对应一个用户，其中存放了用户的账号、密码等个人信息，此外还有是否在线的标记，以及与服务器建立连接的 TCP 套接字。

保存用户信息时，只更新账号和密码信息。对于密码，需要进行加密然后再写入文件，读出文件时使用与之配套的解密算法进行解密。

用户连接：

服务器开放监听套接字，当有用户发送连接请求时，服务器与其建立连接。用户可能执行的操作有：注册、找回密码、登录。对于前两者，套接字在操作完成后就失效，而对于登录的操作，套接字则需要保持长时间连接。因此，当用户建立连接请求的头部为登录信息时，服务器将套接字挂载到用户链表中对应的节点上，并将链表中用户是否在线的标识变量置 1，并获取登录方的 IP 地址，也存到用户结点中。如果用户建立连接请求的头部为注册或找回信息，则在执行完对应功能后销毁套接字。

用户注册：

用户与服务器建立连接，发送注册的账号和密码，服务器到用户链表中查找

是否已经存在该用户，假如不存在，则新建一个用户结点，把该用户添加到用户链表中；否则，发送注册失败的消息，告诉用户，该账号已经被注册了。

离线消息：

对于不在线的用户，无法建立连接，与之聊天时只能把消息发给服务器。发送方在消息头部注明该消息是发给谁的，服务器收到以后，为该用户建立一个文件，把离线消息全都存起来。当离线的用户上线时，服务器检索是否有该用户的离线消息，若有，则打开文件，把消息读出，并发给用户，然后销毁文件，以免占用内存。

2.2 客户端

2.2.1 主要功能介绍

1. 用户注册
2. 找回密码
3. 用户登录
4. 发送和接收文本
5. 发送和接收文件
6. 发送、接收并展示图片表情
7. 获取用户列表
8. 用户界面控制

2.2.2 功能实现

客户端承担聊天的主要功能，包括文本、文件传输，数据包内容构建，界面控制以及与服务器的账号信息交互。

在文本、文件、表情三种信息交互中，文本信息通过 TCP 套接字直接传输，表情信息通过将图片写入字节流后用 TCP 套接字传输。文件传输与图片传输类似，也可以通过 TCP 套接字直接传输字节流，但在学习过 TCP 的流量控制策略后，我尝试基于 UDP 套接字实现文件传输，并手动实现 TCP 的流量控制。

用户注册：

获取用户输入的账号信息，包括账号、密码、密保问题、问题答案，将这些信息打包发给服务器，加上“reg”首部表示是注册信息，等待服务器响应，若服务器返回确认信息，则注册成功，可以使用注册的账号登录，否则，注册账号失败，说明该账号已存在。

找回密码：

用户输入已注册的账号，该账号的密保问题和答案，以及新的密码，将这三项信息加上头部“psd”，打包发给服务器，若服务器返回确认信息，说明找回密码成功，可以使用新密码登录；否则，说明密保问题回答错误，找回密码失败。

用户登录:

将账号和密码, 以及头部“login”发给服务器, 等待服务器响应, 若返回确认信息, 则登录成功, 否则登录失败, 账号与密码不匹配。

获取用户列表:

点击更新联系人按键, 向服务器发送一个请求信息, 头部为“getfriend”, 请求服务器将当前所有已经注册的用户反馈过来, 这样就可以选择其中的用户进行聊天。收到服务器的反馈以后, 建立一个用户链表, 每个结点代表一个用户, 并将每个人的账号显示在列表中。当需要与某人聊天时, 双击即可。

发送和接收文本信息:

在获取了好友列表以后, 在好友列表中点击一位好友, 客户端将向服务器发送获取 IP 地址的请求, 请求服务器把返回选中的用户的 IP 地址。服务器可能有两种响应: 返回用户 IP, 或者返回错误信息。正常返回用户 IP 时, 客户端向该 IP 地址发送 TCP 连接请求, 连接成功后, 在文本框内输入消息, 点击发送, 就可以使用已经建立的 TCP 连接把消息发送过去。假如服务器返回了错误信息, 说明该用户当前不在线, 所以也就不存在 IP 地址一说, 故不能直接和他聊天, 而应该以离线消息的形式发给服务器。在用户看来, 对方在线或者不在线是没有区别的, 消息都成功发送出去了, 但是在内部实现时, 要根据对方是否在线, 决定将消息发向对方或者是发向服务器。

接收信息时, 对于好友发来的消息, 在消息框内进行显示, 并在界面左上角显示当前联系人, 也就是发消息过来的人。对于服务器发来的离线消息, 处理程序类似, 只是在消息显示时, 注明是离线消息, 以便让用户知道这不是当前发的消息。

发送和接收文件:

用户点击获取文件, 弹出文件选择框, 选择待发送的文件, 获取其文件名, 将文件名发送给好友, 并向其发送 UDP 端口请求, 请求对方提供他的 UDP 端口号, 以便进行 UDP 文件传输。对方收到文件名以后, 创建文件, 并反馈端口号, 文件传输开始。发送方一次从文件中读取 1400 个字节的数据, 再加上一个从 1 开始的序号, 用 UDP 套接字发送给对方。在拥塞控制上, 使用了 TCP 的 Reno 算法, 拥塞窗口长度一开始为 1, 然后在每次收到正确的 ack 以后长度翻倍。当收到三个冗余 ack 时, 窗口减半; 当发生超时, 窗口减为 1。发送结束后, 关闭套接字和文件。

接收的文件存储在本地 files 文件夹下, 根据好友发送的文件名, 在本地创建文件, 并将收到的数据写入文件。若收到的序号是递增的, 则将数据写入文件, 并返回递增的 ack 序号。若收到的数据包序号不是当前想要的, 就发送重复的 ack, 直到收到正确的数据包。文件接受完毕后, 关闭文件。

发送、接收并展示表情:

用户点击发送表情, 在本地表情文件夹 img 中选择表情发送, 获取其文件

名，将文件名发送给好友，并用一个字节数组保存图片文件。好友在收到请求后，发送确认包。发送方在收到确认包后将字节数组写入套接字传输给对方，并将表情图片添加到消息列表中。接收方在读取接收缓冲区，并在 `img` 下新建表情，将字节数组写入文件中，并将表情图片添加到消息列表中。另外，对于 gif 形式的表情包，使用 `Qmovie` 对象保存达到播放动图效果。

界面控制：

在不同的用户操作状态下，UI 界面上显示不同的组件。设立一个状态变量，界面控制函数里有一个状态机，它会根据当前所处的状态进行控件的隐藏和显示。

第三章 测试截图及说明

3.1 服务器部分

服务器页面显示用户列表，包括在线用户列表和离线用户列表。

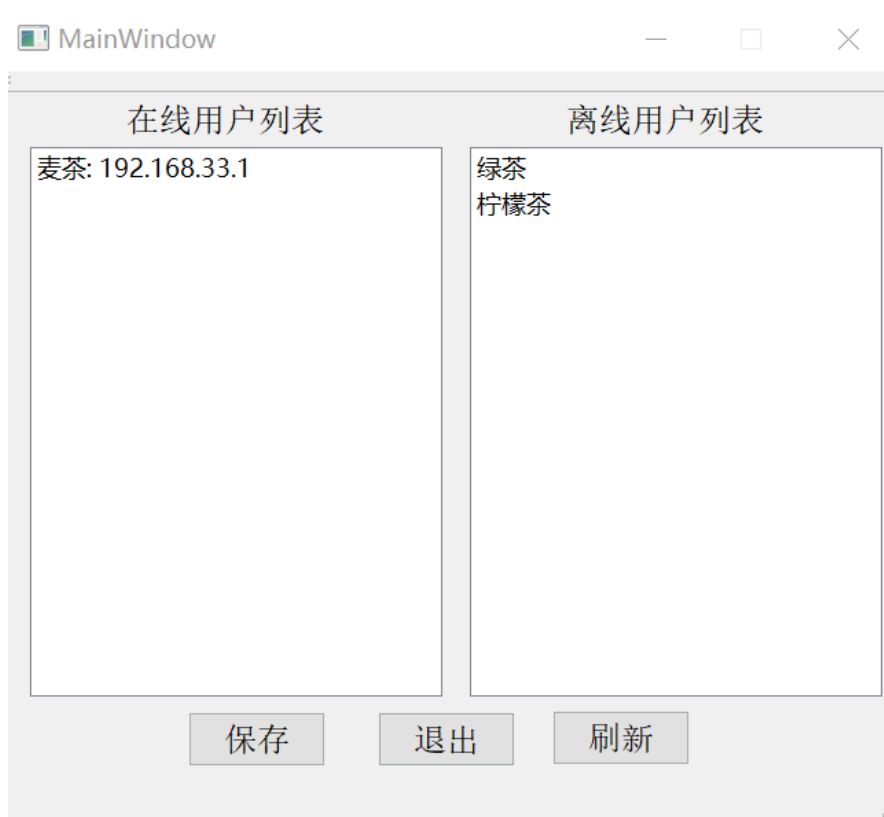


图 3-1 服务器界面展示

3.2 客户端部分

注册界面

新用户填写账号信息进行注册，这里以账号“麦茶”为示例，如图3-2所示：

登录界面

用户填写账号密码信息后登录，上方工具栏包括新用户注册和找回密码功能，如图3-4所示。

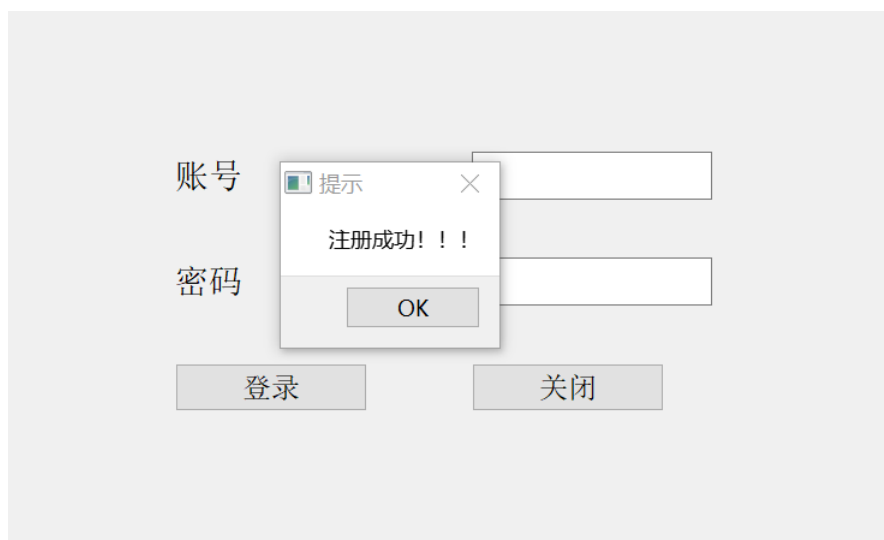
找回密码界面

用户忘记密码时，可通过输入账号、密保问题和答案来更改密码，如图3-5所示。



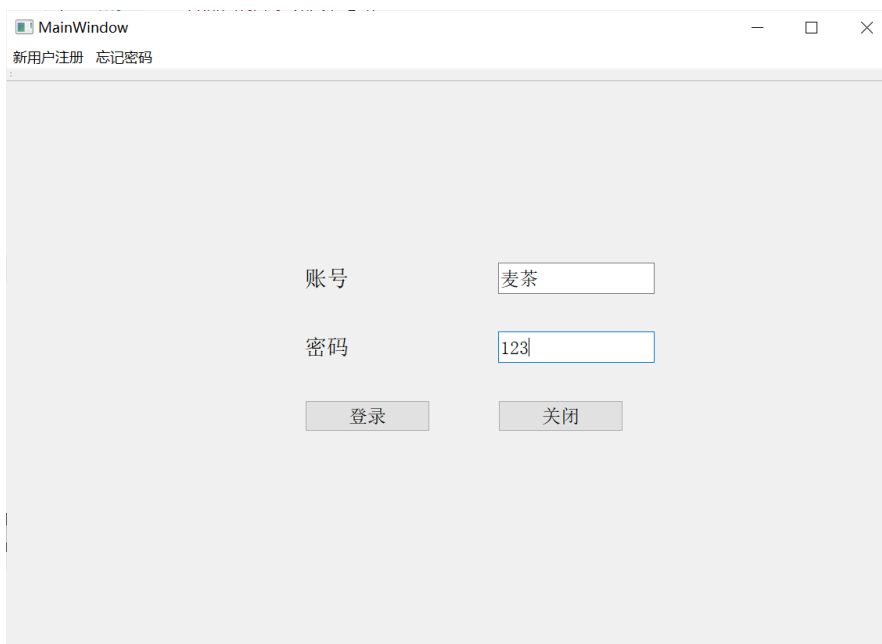
A screenshot of a Qt-based registration window titled "Form". The window contains five input fields with corresponding labels: "账号" (Account) with "麦茶" (Maicha), "密码" (Password) with "123", "再次确认密码" (Confirm Password) with "123", "密保问题" (Security Question) with "你是谁" (Who are you), and "答案" (Answer) with "tea". At the bottom, there are two buttons: "确认" (Confirm) and "取消" (Cancel).

图 3-2 注册界面展示



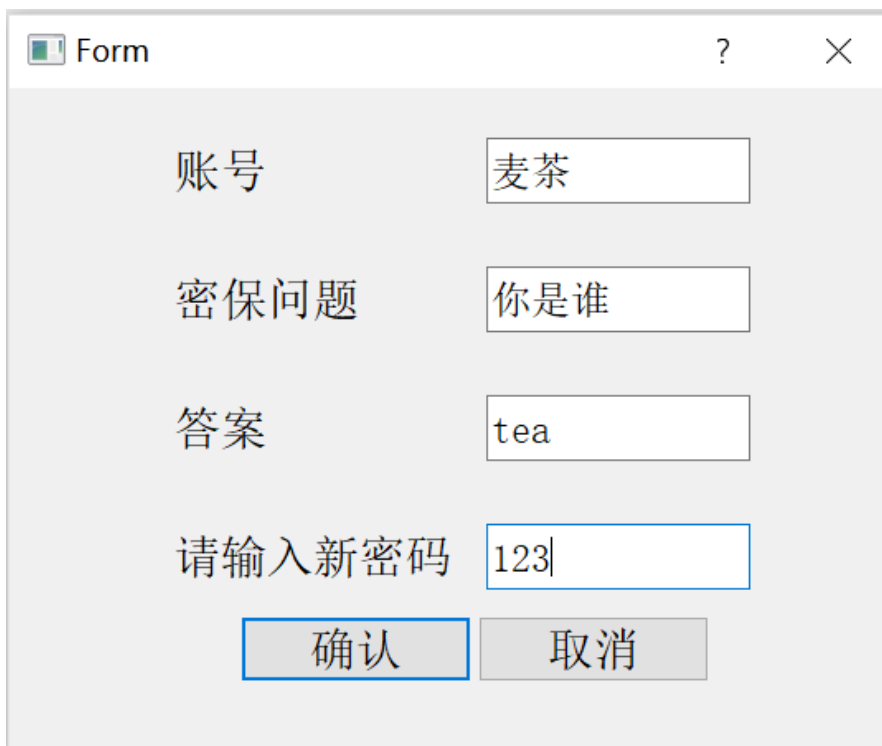
A screenshot showing a registration success prompt dialog box overlaid on a registration form. The dialog box is titled "提示" (Prompt) and contains the text "注册成功!!!" (Registration successful!!!) and an "OK" button. The background form shows labels for "账号" (Account) and "密码" (Password), and buttons for "登录" (Login) and "关闭" (Close).

图 3-3 注册成功提示



The image shows a Qt application window titled "MainWindow". At the top left, there are two links: "新用户注册" (New User Registration) and "忘记密码" (Forgot Password). The main area contains two labels, "账号" (Account) and "密码" (Password), each followed by a text input field. The "账号" field contains the text "麦茶" (Maicha) and the "密码" field contains "123". Below these fields are two buttons: "登录" (Login) and "关闭" (Close).

图 3-4 登录界面展示



The image shows a Qt application window titled "Form". It contains four labels with corresponding text input fields: "账号" (Account) with "麦茶" (Maicha), "密保问题" (Security Question) with "你是谁" (Who are you), "答案" (Answer) with "tea", and "请输入新密码" (Please enter new password) with "123". At the bottom, there are two buttons: "确认" (Confirm) and "取消" (Cancel).

图 3-5 修改界面展示

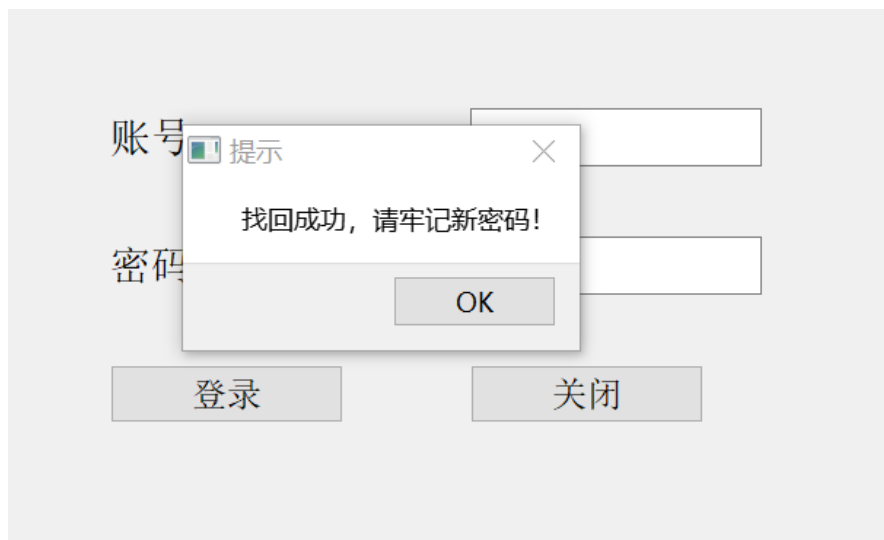


图 3-6 修改密码成功提示

聊天主界面

用户登录成功后，进入聊天主界面。界面包括文件传输列表，用户列表，消息列表。消息列表展示当前的 IP 地址，TCP 端口号和 UDP 端口号，如图3-7所示。



图 3-7 聊天主界面

文字聊天

用户在更新联系人后，从联系人列表中双击好友开始聊天，此时左上角显示当前聊天对象，消息列表中显示“newconnection”，如图3-8所示。

多组用户聊天

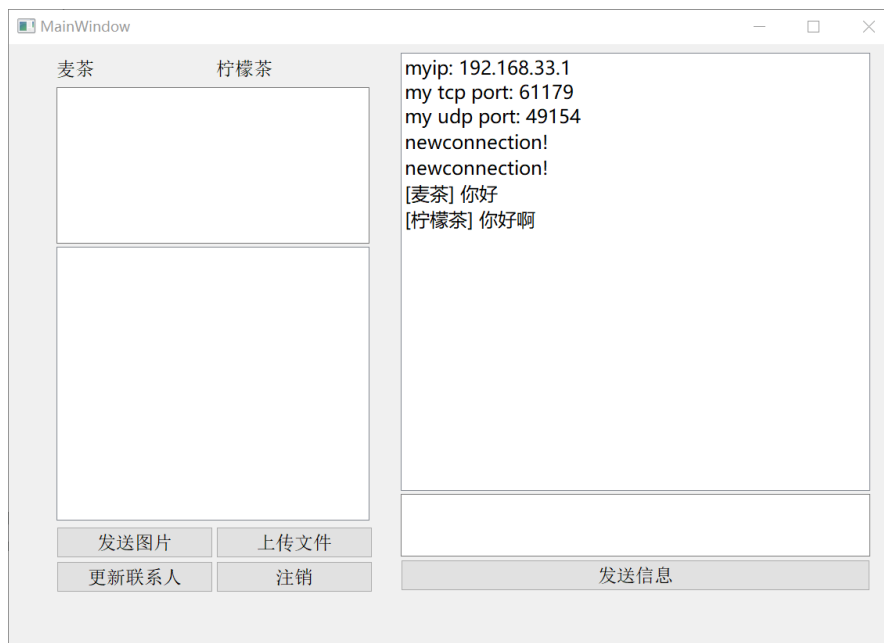


图 3-8 文字聊天展示

当 2 个用户正在聊天时，其他在线用户可能向他们发送信息，此时他们应该能正常接收其他用户的信息。这里假设柠檬茶和绿茶正在聊天，麦茶向柠檬茶发送消息，如图3-9所示。

离线消息

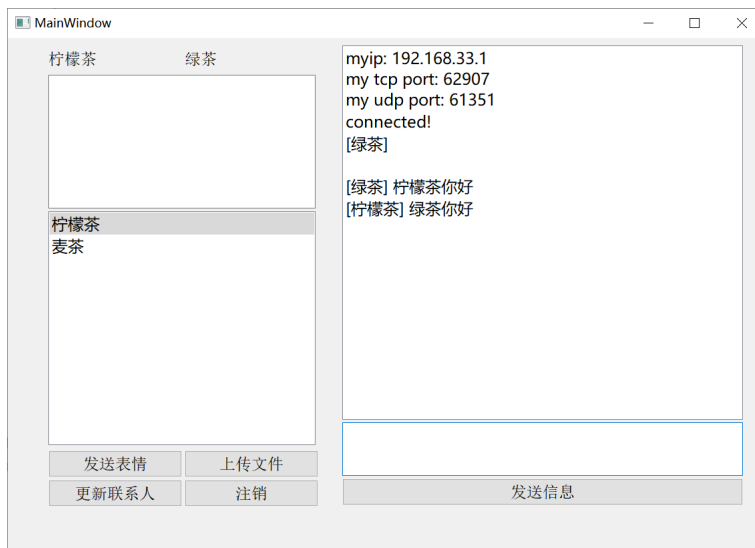
当目标用户不在线时，消息会发送给服务器保存，待用户上线后由服务器发送给目标用户，如图3-10所示。

文件传输

一方向另一方发送文件，文件传输列表中会显示文件路径和传输时间，如图3-11所示，文件存储在 files 文件夹下，图3-12展示了文本文档、图片、pdf、可执行文件的传输效果。

发送表情

用户发送表情展示，表情存储在 img 文件夹下，支持 png, jpg, jpeg, gif 四种形式，如图3-13所示。



a) 柠檬茶和绿茶正在聊天



b) 麦茶向柠檬茶发送消息

图 3-9 多组用户聊天展示



a) 发送离线消息



b) 登录后收到离线消息

图 3-10 离线消息展示



a) 发送文件



b) 接收文件

图 3-11 传输文件



图 3-12 传输文件类型



a) 发送表情



b) 接收表情

图 3-13 发送表情

第四章 项目总结

本次大作业中，我以开发聊天程序为选题，一是因为个人对软件开发比较感兴趣，二是希望能积累客户端与服务器交互的开发经验。该项目的主要任务是利用 socket 接口，通过传输层协议实现各种数据类型的传输，难点主要在于协调套接字通信，文件读写与格式转换，客户端 UI 设计等。在一个多月的开发中，我实现了用户与服务器间的基础信息交互，以及用户间发送文本、表情、文件等功能，最后，为了使实验聊天软件更接近市面上的商业聊天软件，我又补充了账号密码，密保等安全功能，并实现了找回密码功能。

项目开展过程可谓困难重重。尤其在项目前期，我要在多种实现方法中选择一个来完成，一开始，我准备以 web 形式完成，先后尝试了 golang 和 Python 的后台服务器，但在使用 websocket 进行文件传输时，却遇到了瓶颈。在尝试了几天无果后，我又转向客户端和服务器的结构，重新学习 Qt 来设计。面对文件传输方法，在查阅大量资料和与同学沟通后，我最终通过将文件内容读入字节数组，用套接字传输字节数组成功实现。而在学过传输层的 TCP 和 UDP 协议的区别后，我又尝试基于 UDP 协议传输文件，手动实现 TCP 的流量控制，加强了我对 TCP 协议的理解。最终，在这次项目中，我将客户端对客户端，客户端对服务器，TCP 套接字和 UDP 套接字等多种方法结合在一起，使涉及的知识更加全面，进一步锻炼了自己的能力。

通过此次大作业的实战，我受益良多，不仅提高了我的编程能力，学习了套接字编程方法，而且对传输层知识有了更加深刻的理解。感谢老师和同学提供的指导和帮助！

附录 A 部分源代码

A.1 客户端

A.1.1 主页面

```
#include "mainwindow.h"
#include "registerdialog.h"
#include "findpswdialog.h"
#include "ui_mainwindow.h"
#include <QDebug>

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent), ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    winMax = 100;
    winThreshold = 32;
    winLen = 1; // 阻塞窗口长度
    end = winLen;
    rtt = 1;
    timers = new QTimer[winMax+1];
    arrbuf = new QByteArray[winMax+1];
    reg_dialog = 0;
    find_dialog = 0;
    fileDialog = 0;
    conn_or_not = 0;
    chatterState = 0;
    msg_file = 1;
    allusers = new user();
    conn_users = new login();
    conn_users->next = 0;
    sequenceAccepted = 0;
    sequenceSent = 0;
    segLength = 1400;
    tail = 0;
    // tail = 0x7fffffffffffffff;
    state = 0;
    senderPort = receiverPort = 0;
    serverip = QString("127.0.0.1"); // 本地服务器
    this->ui->msg_list->setIconSize(QSize(200, 200));

    connect(this->ui->actionregister, &QAction::triggered, this, &MainWindow::on_action_register_triggered);
    connect(this->ui->actionIP, &QAction::triggered, this, &MainWindow::on_action_IP_triggered);
    connect(this->ui->actionfoget, &QAction::triggered, this, &MainWindow::on_action_find_triggered);
    connect(this->ui->btn_send, &QPushButton::clicked, this, &MainWindow::on_btn_send_clicked);
    connect(this->ui->btn_getfriend, &QPushButton::clicked, this, &MainWindow::on_btn_getfriend_clicked);
    connect(this->ui->btn_file, &QPushButton::clicked, this, &MainWindow::on_btn_file_clicked);
    connect(this->ui->btn_img, &QPushButton::clicked, this, &MainWindow::sending);

    for(int i=0; i<=winMax; ++i)
        connect(timers+i, SIGNAL(timeout()), this, SLOT(on_resend())); // 传输超时重发
    this->m_socket = new QTcpSocket(this); // 与服务器的 socket
    this->chat_socket = new QTcpSocket(this); // 与聊天对象的 socket
    this->current_socket = new QTcpSocket(this); // 当前 socket, 变量
    this->chat_server = new QTcpServer(this); // 当前 TCP 的连接口
    this->udpreceiver = new QUdpSocket(this); // 传输文件用
    this->udpender = new QUdpSocket(this);

    chat_server->listen(QHostAddress::Any, 0);
    myport = chat_server->serverPort();

    connect(this->m_socket, SIGNAL(connected()), this, SLOT(on_connected()));
```

```

connect(this->m_socket, SIGNAL(disconnected()), this, SLOT(on_discon()));
connect(this->chat_server, SIGNAL(newConnection()), this, SLOT(chat_connected()));
connect(this->chat_socket, SIGNAL(connected()), this, SLOT(chatting_connected()));
logout_succeed();
QString localHostName = QHostInfo::localHostName(); //Returns this machine's host name
QHostInfo info = QHostInfo::fromName(localHostName);
foreach(QHostAddress address, info.addresses())
{
    if(address.protocol() == QAbstractSocket::IPv4Protocol)
        localip = address.toString();
}
QDebug()<<localip<<myport; //show IP and Port

hostReceiver.setAddress(localip);
udpreceiver->bind(hostReceiver, receiverPort); //设置本地ip地址的端口
receiverPort = udpreceiver->localPort();

connect(udpreceiver, SIGNAL(readyRead()), this, SLOT(readPendingDatagrams()));
QDebug()<<udpserver->localAddress()<<udpserver->localPort()<<senderPort;
QDebug()<<udpreceiver->localAddress()<<udpreceiver->localPort()<<receiverPort;
}
MainWindow::~MainWindow(){ delete ui;}
void MainWindow::chatting_connected(){
    chatterState = 1;
    current_socket = chat_socket;
    connect(this->chat_socket, SIGNAL(readyRead()), this, SLOT(on_readychatting()));
    connect(this->chat_socket, SIGNAL(disconnected()), this, SLOT(chatting_dis()));
    ui->msg_list->addItem("connected!");
}
void MainWindow::readPendingDatagrams()
{
    while(udpreceiver->hasPendingDatagrams())
    {
        arr.resize(udpreceiver->pendingDatagramSize());
        udpreceiver->readDatagram(arr.data(), arr.size(), &host, &port);
        QDataStream readdst(&arr, QIODevice::ReadWrite);
        readdst>>sequencetemp>>dataarr;
        if(dataarr.size()) //dataarr非0, 接受文件
        {
            sequenceSent = sequencetemp;
            QDataStream writedst(&arr, QIODevice::ReadWrite);
            arr.clear();
            if(sequenceSent == sequenceAccepted + 1) //若收到的序号是递增的, 则将数据写入文件, 并返回递增的ack
            {
                sequenceAccepted++;
                file_rw->write(dataarr.data(), dataarr.size());
            }
            else if(sequencetemp == 0)
            {
                if(file_rw)
                {
                    ui->file_show->append("文件接收完毕!");
                    file_rw->close();
                    delete file_rw;
                    file_rw = 0;
                }
                sequenceAccepted = 0;
            }
            // else
            // ui->file_show->append(QString::number(sequencetemp));
            writedst<<sequenceAccepted;
            udpserver->writeDatagram(arr, chatterHost, chatter_udpport); //Sends the datagram
        }
        else //dataarr为0, 上传文件
        {
            if(sequencetemp==sequenceAccepted + 1)
            {

```



```

sequenceAccepted++;
state = 0;
timers[sequenceAccepted % winMax].stop();
if(sequenceAccepted == end) {
    if(winLen < winThreshold)
        winLen *= 2;
    else if(winLen < winMax)
        winLen += 1;
    end = end + winLen;
    send();
}
}
else if(sequenceTemp == 0)
{
    sequenceSent = 0;
    sequenceAccepted = 0;
    state = 0;
    if(file_rw) {
        ui->file_show->append("文件发送完毕!");
        file_rw->close();
        delete file_rw;
        file_rw = 0;
        //tail = 0x7fffffffffffffff;
        tail = 0;
        endup = QDateTime::currentDateTime();
        ui->file_show->append("传输用时: "+QString::number(start.secsTo(endup))+"秒");
    }
}
}
}
}

void MainWindow::on_resend()
{
    winThreshold /= 2;
    winLen = winThreshold + 3;
    reSend();
}

void MainWindow::reSend() // 存在问题
{
    for(int i=sequenceAccepted+1;i<=end; ++i)
        timers[i%winMax].stop();
    for(int i=sequenceAccepted + 1;i<=end ;++i)
    {
        //if(i<=tail)
        if((!tail) || (i<=tail))
        {
            sequenceSent = i;
            udpSender->writeDatagram(arrbuf[sequenceSent % winMax], chatterHost, chatter_udpport);
            timers[sequenceSent%winMax].start(rtt);
        }
    }
}

void MainWindow::send()
{
    for(int i=sequenceAccepted + 1;i<=end ;++i)
    {
        sequenceSent = i;
        if(! file_rw->atEnd()) {
            QDataStream writedst(arrbuf+sequenceSent%winMax, QIODevice::ReadWrite);
            arrbuf[sequenceSent%winMax].clear();
            writedst<<sequenceSent;
            writedst<<file_rw->read(segLength);
            udpSender->writeDatagram(arrbuf[sequenceSent%winMax], chatterHost, chatter_udpport);
        }
        else {
            QDataStream writedst(arrbuf+sequenceSent%winMax, QIODevice::ReadWrite);
            arrbuf[sequenceSent%winMax].clear();

```

```
writedst << qint64(0) << "end";
udpSender->writeDatagram(arrbuf[sequenceSent%winMax].data(),
    arrbuf[sequenceSent%winMax].size(), chatterHost, chatter_udpport);
tail = sequenceSent;
break;
}
timers[sequenceSent%winMax].start(rtt);
}
}

void MainWindow::sendFile()
{
    file_rw = new QFile(file_full);
    file_rw->open(QIODevice::ReadOnly);
    sequenceSent=0;
    sequenceAccepted = 0;
    winThreshold = 32;
    winLen = 1;
    ackcount = 0;
    end = winLen + sequenceAccepted;
    //t.start();
    send();
}

void MainWindow::receiveFile()
{
    file_rw = new QFile(file_name);
    file_rw->open(QIODevice::WriteOnly|QIODevice::Append|QIODevice::Unbuffered);
    file_rw->resize(0);
    sequenceAccepted = 0;
}

void MainWindow::chatting_dis(){
    chatterState = 0;
}

void MainWindow::chatted_connected()
{
    chatterState = 1;
    ui->msg_list->addItem("newconnection!");
    login* p = new login(chat_server);
    p->next = conn_users->next;
    conn_users->next = p;
    connect(p, SIGNAL(newMessage(login*)), this, SLOT(on_readyread(login*)));
    current_socket = p->socket;
}

void MainWindow::on_readychatting()
{
    current_socket = chat_socket;
    chatterState = 1;
    handleMsg();
}

void MainWindow::on_readyread(login* sender)
{
    current_socket = sender->socket;
    handleMsg();
    ui->label_friend->setText(current_chatter);
    sender->account = current_chatter;
}

void MainWindow::handleMsg() // 处理收到的信息
{
    QByteArray arr;
    arr = current_socket->readAll();
    QDataStream dst(&arr, QIODevice::ReadWrite);
    QString head, my_account, chatter, str;
    dst>>head>>chatter;
    if(chatter.size())
    {
        current_chatter = chatter;
        if(head=="chatmsg")
        {

```

```
dst>>my_account>>str;
ui->label_friend->setText(current_chatter);
ui->msg_list->addItem("[ "+current_chatter+" ]_"+str);
}
else if(head=="filepath")
{
    dst>>file_name>>chatterHost>>chatter_udpport;
    ui->file_show->append("[ "+current_chatter+" ]_"+str);
    dialogSave();
}
else if(head=="filewanted")// 文件名已收到, 可发送文件
{
    msg_file = 1;
    dst>>chatter_udpip>>chatter_udpport;
    chatterHost.setAddress(chatter_udpip);
    ui->file_show->append("开始发送文件!");
    start = QDateTime::currentDateTime();
    sendFile();
}
else if(head=="img")
{
    qDebug()<<"img";
    int imgSize;
    QString imgName;
    dst>>imgName>>imgSize;
    disconnect(current_socket, SIGNAL(readyRead()), this, SLOT(on_readychatting()));
    arr.clear();
    QDataStream wdst(&arr, QIODevice::ReadWrite);
    wdst<<QString("accepted");
    sendMessage(arr, current_socket);
    QByteArray contentByteArray, subContentByteArray;
    while (imgSize > 0)
    {
        if (current_socket->waitForReadyRead())
        {
            subContentByteArray = current_socket->readAll();
            contentByteArray.append(subContentByteArray);
            imgSize -= subContentByteArray.size();
        }
    }
    connect(current_socket, SIGNAL(readyRead()), this, SLOT(on_readychatting()));
    if (imgName.split('.').last() == "gif")
    {
        QDir dir(QDir::currentPath());
        if (!dir.exists("img"))
        {
            dir.mkdir("img");
        }
        imgName = QString("%1/%2").arg("img", imgName);
        QFile imgFile(imgName);
        if (!imgFile.open(QIODevice::WriteOnly))
        {
            return;
        }
        imgFile.write(contentByteArray);
        QLabel *gifLabel = new QLabel;
        QMovie *gif = new QMovie(imgName);
        gif->setScaledSize(QSize(200, 200));
        if (!gif->isValid())
        {
            delete gifLabel;
            return;
        }
        gifLabel->setMovie(gif);
        QListWidgetItem *item = new QListWidgetItem;
        item->setSizeHint(QSize(200, 200));
        ui->msg_list->addItem("[ "+current_chatter+" ]");
    }
}
```



```

        ui->msg_list->addItem(item);
        ui->msg_list->setItemWidget(item, gifLabel);
        gif->start();
    }
    else
    {
        QPixmap pixmap;
        pixmap.loadFromData(contentByteArray);
        QIcon img(pixmap);
        QListWidgetItem *item = new QListWidgetItem(img, "");
        ui->msg_list->addItem("[ "+current_chatter+" ]");
        ui->msg_list->addItem(item);
    }
}
}
}
}
void MainWindow::dialogSave()
{
    QMessageBox box;
    box.question(this, "文件提醒", "收到新文件: "+file_name+"_是否保存?",
        QMessageBox::Yes | QMessageBox::No, QMessageBox::Yes);

    if (box.Accepted)
    {
        arr.clear();
        QDataStream wdst(&arr, QIODevice::ReadWrite);
        wdst<<QString("filewanted")<<myaccount<<localip<<receiverPort;
        sendmessage(arr, current_socket);
        ui->file_show->append("准备接收文件 ... ..\n");
        receiveFile();
    }
}
void MainWindow::on_btn_send_clicked()
{
    if (current_chatter.size())
    {
        QString str=this->ui->text_send->toPlainText();
        if (str.size())
        {
            this->ui->text_send->clear();
            this->arr.clear();
            QDataStream dst(&arr, QIODevice::ReadWrite);
            if (msg_file) // 发送文本信息
            {
                dst<<QString("chatmsg") <<myaccount<<current_chatter<<str;
                this->ui->msg_list->addItem("[ "+myaccount+" ]_"+str);
            }
            else // 发送文件
            {
                dst<<QString("filepath") <<myaccount<<file_name<<hostReceiver<<receiverPort;
                this->ui->file_show->append("[ "+myaccount+" ]_"+file_full);
                ui->file_show->append("等待对方确认接收文件 ... ..");
            }
            if (chatterState) // 判断对方是否在线
                sendmessage(arr, current_socket);
            else sendmessage(arr);
        }
    }
}
void MainWindow::reconnect(){
    this->m_socket->connectToHost(serverip, 2001, QTcpSocket::ReadWrite);
}
void MainWindow::connect_to_chatter(){
    this->chat_socket->connectToHost(chatter_ip, chatter_port, QTcpSocket::ReadWrite);
}
void MainWindow::on_connected(){
    conn_or_not = 1;
    connect(this->m_socket, SIGNAL(readyRead()), this, SLOT(on_readyread()));
}

```

```
}  
void MainWindow::on_discon(){  
    conn_or_not = 0;  
    logout_succeed();  
}  
  
void MainWindow::on_list_friend_itemDoubleClicked(QListWidgetItem *item)  
{  
    current_chatter = item->text();  
    ui->label_friend->setText(current_chatter);  
    arr.clear();  
    QDataStream dst(&arr,QIODevice::ReadWrite);  
    dst<<QString("applyforip")<<myaccount<<current_chatter;  
    sendmessage(arr);  
}  
  
void MainWindow::on_btn_file_clicked()  
{  
    fileDialog = new QFileDialog(this); // 选择文件  
    fileDialog->setWindowTitle(tr("选择文件")); // 定义文件对话框标题  
    // 设置可以选择多个文件, 默认为只能选择一个文件 QFileDialog::ExistingFiles  
    fileDialog->setFileMode(QFileDialog::ExistingFiles);  
    fileDialog->setAcceptMode(QFileDialog::AcceptOpen);  
    fileDialog->setViewMode(QFileDialog::Detail); // 设置视图模式  
    ui->text_send->clear();  
    if (fileDialog->exec()==QDialog::Accepted)  
    {  
        msg_file = 0;  
        file_full = fileDialog->selectedFiles().at(0);  
        ui->text_send->append("[ 文件 ]_" + file_full);  
        QFileInfo info = QFileInfo(file_full);  
        file_name = info.fileName();  
    }  
    delete fileDialog;  
}  
  
void MainWindow::on_readyread()  
{  
    QByteArray arr=this->m_socket->readAll();  
    QDataStream dst(&arr,QIODevice::ReadWrite); // 重点  
    QString head,head2;  
    dst>>head;  
    QString account;  
    if (head=="reg")  
    {  
        dst>>head2>>account;  
        if (account==this->reg_dialog->account)  
        {  
            if (head2=="ack")  
            {  
                reg_dialog->close();  
                delete reg_dialog;  
                m_socket->close();  
                QMessageBox::about(this,"提示","注册成功!!!");  
            }  
            else if (head2=="fail")  
                QMessageBox::about(this,"警告","注册失败, 账号已存在!!!");  
        }  
    }  
    else if (head=="login")  
    {  
        dst>>head2>>account;  
        if (account==this->account)  
        {  
            if (head2=="ack")  
            {  
                login_succeed();  
                QString off_or_not,msg,chatter;  
                dst>>off_or_not;  
                if (off_or_not=="offmsg") // 有离线消息
```



```

        {
            qDebug() << "离线";
            while (1)
            {
                dst >> chatter >> msg;
                if (chatter.size())
                {
                    ui->msg_list->addItem(QString(chatter+"_"+"(离线消息)"+msg));
                }
                else break;
            }
        }
    }
    else if (head2 == "fail")
    {
        QMessageBox::about(this, "警告", "登录失败, 账号或密码错误!!!");
    }
    else if (head2 == "none")
    {
        QMessageBox::about(this, "警告", "登录失败, 账号不存在!!!");
    }
}

else if (head == "find")
{
    dst >> head2 >> account;
    if (account == this->find_dialog->account)
    {
        if (head2 == "ack")
        {
            find_dialog->close();
            delete find_dialog;
            m_socket->close();
            QMessageBox::about(this, "提示", "找回成功, 请牢记新密码!");
        }
        else if (head2 == "fail")
        {
            QMessageBox::about(this, "警告", "找回失败, 问题回答错误!!!");
        }
        else if (head2 == "wrong")
        {
            QMessageBox::about(this, "警告", "找回失败, 密保问题错误!!!");
        }
        else
        {
            QMessageBox::about(this, "警告", "找回失败, 账号不存在!!!");
        }
    }
}

else if (head == "friend")
{
    allusers->destroy();
    user *p;
    int user_num;
    dst >> user_num;
    ui->list_friend->clear();
    for (int i = 0; i < user_num; ++i)
    {
        p = new user();
        dst >> p->account;
        p->next = allusers->next;
        allusers->next = p;
        ui->list_friend->addItem(p->account);
    }
}

else if (head == "getip")
{
    chat_socket->close(); // 先断开, 再连接, 防止重复连接
    dst >> head2;
    if (head2 == "friendonline")
    {
        dst >> chatter_ip >> chatter_port;
        chatterState = 1;
        login *p = conn_users->find(current_chatter); // 不安全
        if (p)
        {
            if (p->inout)
            {

```

```

        // QMessageBox::about(this, "x", "chatted before");
        current_socket = p->socket;
        return;
    }
}
connect_to_chatter();
}
else if(head2=="friendoff")
{
    chatterState = 0;
}
}
}

void MainWindow::on_btn_close_clicked() { this->close(); }

void MainWindow::sending() {
    QDir dir;
    dir.currentPath();
    if (!dir.exists("img"))
    {
        dir.mkdir("img");
    }
    dir.cd("img");
    QString imgPath = QFileDialog::getOpenFileName(this, "发送本地表情",
                                                    dir.currentPath() + "/img", "*.png;*.jpg;*.jpeg;*.gif");

    if (imgPath == "")
    {
        return;
    }
    QFile img(imgPath);
    img.open(QIODevice::ReadOnly);
    if (!img.isOpen())
    {
        QMessageBox::about(this, "本地表情", "获取资源失败");
        return;
    }
    int imgSize = img.size();
    QByteArray content = img.readAll();
    img.close();

    if (current_chatter.size() && chatterState) // 判断用户在线，发送请求
    {
        this->arr.clear();
        QDataStream dst(&arr, QIODevice::ReadWrite);
        dst<<QString("img")<<myaccount<<imgPath.split('/').last()<<imgSize; // head, account, name, size
        current_socket->write(arr);
    }
    else
    {
        QMessageBox::about(this, "本地表情", "未建立连接");
        return;
    }
}

disconnect(current_socket, SIGNAL(readyRead()), this, SLOT(on_readychatting()));
if (current_socket->waitForReadyRead())
{
    qDebug()<<"Point2";

    QByteArray arr;
    arr = current_socket->readAll();
    QDataStream dst(&arr, QIODevice::ReadWrite);
    QString head;
    dst>>head;
    qDebug()<<head;
    if (head == "accepted")
    {
        current_socket->write(content);
    }
}

```

```

    }
    else
    {
        QMessageBox::about(this, "本地表情", "接收响应失败");
        connect(current_socket, SIGNAL(readyRead()), this, SLOT(on_readychatting()));
        return;
    }
}

QDebug()<<"Point1";
connect(current_socket, SIGNAL(readyRead()), this, SLOT(on_readychatting()));

QString imgName = imgPath.split('/').last();
if (imgName.split('.').last() == "gif")
{
    QLabel *gifLabel = new QLabel;
    QMovie *gif = new QMovie(imgPath);
    gif->setScaledSize(QSize(200, 200));
    if (!gif->isValid())
    {
        delete gifLabel;
        return;
    }
    gifLabel->setMovie(gif);
    QListWidgetItem *item = new QListWidgetItem;
    item->setSizeHint(QSize(200, 200));
    ui->msg_list->addItem("[ "+myaccount+" ]");
    ui->msg_list->addItem(item);
    ui->msg_list->setItemWidget(item, gifLabel);
    gif->start();
}
else
{
    QPixmap pixmap;
    pixmap.loadFromData(content);
    QIcon img(pixmap);
    QListWidgetItem *item = new QListWidgetItem(img, "");
    ui->msg_list->addItem("[ "+myaccount+" ]");
    ui->msg_list->addItem(item);
}

}

void MainWindow::sendmessage(QByteArray &arr, QTcpSocket *chatter){
    chatter->write(arr);
}

void MainWindow::sendmessage(QByteArray &arr){ //对方离线，发送给服务器
    if(!conn_or_not) reconnect();
    m_socket->write(arr);
}

void MainWindow::on_btn_login_clicked()
{
    account = ui->line_account->text();
    psw = ui->line_psw->text();
    myaccount = account;
    arr.clear();
    QDataStream dst(&arr, QIODevice::ReadWrite);
    dst<<QString("login") <<localip <<myport<<account<<psw;
    sendmessage(arr);
}

void MainWindow::login_succeed()
{
    ui->btn_send->show();
    ui->btn_off->show();
    ui->btn_file->show();
    ui->btn_img->show();
    ui->btn_login->hide();
}

```

```

        ui->btn_close->hide();

        ui->msg_list->show();

        ui->line_account->hide();
        ui->line_psw->hide();

        ui->menuBar->hide();
        ui->mainToolBar->hide();

        ui->label_account->hide();
        ui->label_psw->hide();
        ui->label_friend->show();
        ui->label_myname->show();
        ui->list_friend->show();
        ui->text_send->show();
        ui->msg_list->clear();
        ui->btn_getfriend->show();
        ui->label_myname->setText(myaccount);
        ui->file_show->show();
        ui->file_show->clear();
        ui->msg_list->addItem("myip:_" + localip+"\\nmy_tcp_port:_" +QString::number(myport)+"\\nmy_udp_port:_"
                               +QString::number(receiverPort));
    }
    void MainWindow::logout_succeed()
    {
        ui->btn_send->hide();
        ui->btn_off->hide();
        ui->btn_file->hide();
        ui->btn_img->hide();
        ui->btn_login->show();
        ui->btn_close->show();
        ui->msg_list->hide();
        ui->line_account->show();
        ui->line_psw->show();

        ui->menuBar->show();
        ui->mainToolBar->show();

        ui->label_account->show();
        ui->label_psw->show();
        ui->label_friend->hide();
        ui->label_myname->hide();

        ui->list_friend->hide();
        ui->text_send->hide();
        ui->line_account->clear();
        ui->line_psw->clear();
        ui->btn_getfriend->hide();
        ui->list_friend->clear();
        ui->label_friend->clear();
        ui->file_show->hide();
        current_chatter.clear();
        myaccount.clear();
    }
    void MainWindow::on_action_IP_triggered()
    {
        setipwindow = new SetIPDialog();
        setipwindow->show();
        connect(setipwindow, SIGNAL(emitIP(QString &)), this, SLOT(setServerIP(QString &)));
    }
    void MainWindow::setServerIP(QString &ip)
    {
        serverip = ip;
        delete setipwindow;
        ui->menu_IP->hide();
    }

```

```
void MainWindow::on_action_register_triggered()
{
    reg_dialog = new RegisterDialog;
    reg_dialog->show();
    connect(this->reg_dialog, SIGNAL(sendregmsg(QByteArray &)),
            this, SLOT(sendmessage(QByteArray &)));
}

void MainWindow::on_action_find_triggered()
{
    find_dialog = new findpswDialog;
    find_dialog->show();
    connect(this->find_dialog, SIGNAL(sendfindmsg(QByteArray &)),
            this, SLOT(sendmessage(QByteArray &))); // 是否应该在别处连接
}

void MainWindow::on_btn_off_clicked()
{
    arr.clear();
    QDataStream dst(&arr, QIODevice::ReadWrite);
    dst<<QString("logout") <<myaccount;
    sendmessage(arr);
    m_socket->close();
    chat_socket->close();
    conn_users->destroy();
    logout_succeed();
}

void MainWindow::on_btn_getfriend_clicked()
{
    arr.clear();
    QDataStream dst(&arr, QIODevice::ReadWrite);
    dst<<QString("getfriend") <<myaccount;
    sendmessage(arr);
}
```

A.1.2 找回密码

```
#include "findpswdialog.h"
#include "ui_findpswdialog.h"
```

```
findpswDialog::findpswDialog(QWidget *parent) :
    QDialog(parent), ui(new Ui::findpswDialog)
{
    ui->setupUi(this);

    connect(this->ui->btn_ok, &QPushButton::clicked, this, &findpswDialog::on_btn_ok_clicked);
    connect(this->ui->btn_cancel, &QPushButton::clicked, this, &findpswDialog::on_btn_cancel_clicked);
}

findpswDialog::~findpswDialog()
{
    delete ui;
}

void findpswDialog::on_btn_cancel_clicked()
{
    this->close();
    this->~findpswDialog();
}

void findpswDialog::on_btn_ok_clicked()
{
    account = this->ui->line_account->text();
    question = this->ui->line_question->text();
    answer = this->ui->line_answer->text();
    psw = this->ui->line_psw->text();
    arr.clear();
    QDataStream dst(&arr, QIODevice::ReadWrite);
    dst<<QString("find")<<account<<question<<answer<<psw;
```

```
emit sendfindmsg(arr);  
}
```

A.1.3 登录

```
#include "login.h"  
  
login::login(QTcpServer*server)  
{  
    next =0;  
    inout =1;  
    valid = 0;  
    socket = new QTcpSocket(this);  
    socket = server->nextPendingConnection();  
    connect(socket,SIGNAL(readyRead()),this,SLOT(on_connected()));  
    connect(socket,SIGNAL(disconnected()),this,SLOT(on_disconned()));  
    qDebug()<<socket->peerName()<<socket->peerAddress()<<socket->peerPort();  
}  
login::login()  
{  
    next =0;  
    inout =0;  
    valid = 0;  
    socket = 0;  
}  
void login:: on_connected()  
{  
    emit newMessage(this);  
}  
void login::on_disconned()  
{  
    inout = 0;  
}  
login* login::find(QString &account)  
{  
    login *p = this->next;  
    for(; p; p=p->next)  
    {  
        if(p->account==account)  
            return p;  
    }  
    return 0;  
}  
void login:: destroy()  
{  
    login*p = this->next, *pp=p;  
    while(pp)  
    {  
        pp=p->next;  
        delete p;  
        p = pp;  
    }  
    this->next = 0;  
}  
login::~~login()  
{  
    if(socket)  
        delete socket;  
}
```

A.1.4 注册

```
#include "registerdialog.h"  
#include "ui_registerdialog.h"
```

```
RegisterDialog::RegisterDialog(QWidget *parent) :
    QDialog(parent), ui(new Ui::RegisterDialog)
{
    ui->setupUi(this);
    connect(this->ui->btn_ok,&QPushButton::clicked,this,&RegisterDialog::on_push_ok_clicked);
    connect(this->ui->btn_cancel,&QPushButton::clicked,this,&RegisterDialog::on_push_cancel_clicked);
}
RegisterDialog::~RegisterDialog(){
    delete ui;
}
void RegisterDialog::on_push_ok_clicked()
{
    psw = this->ui->line_psw->text();
    QString psw2 = this->ui->line_psw2->text();
    if(psw != psw2)
    {
        QMessageBox::about(this,"警告","两次输入密码不一致!!!");
        return;
    }
    account=this->ui->line_account->text();
    question=this->ui->line_question->text();
    answer=this->ui->line_answer->text();
    arr.clear();
    QDataStream dst(&arr,QIODevice::ReadWrite);
    dst<<QString("reg")<<account<<psw<<question<<answer;
    emit sendregmsg(arr);
}
void RegisterDialog::on_push_cancel_clicked()
{
    this->close();
    this->~RegisterDialog();
}
```

A.1.5 用户类

```
#include "user.h"

user::user()
{
    next = 0;
}
void user::destroy()
{
    user*p = this->next, *pp=p;
    while(pp)
    {
        pp=p->next;
        free(p);
        p = pp;
    }
    this->next = 0;
}
```

A.2 服务端

A.2.1 主页面

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QHostAddress>
#include <cstdio>
#include <QDebug>
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent), ui(new Ui::MainWindow)
```

```
{
    ui->setupUi( this );
    user_num = 0;
    client_num = 0;
    port_number = 1997;
    file = new QFile("Register.txt");

    connect( this->ui->btn_exit, &QPushButton::clicked, this, &MainWindow::on_btn_exit_clicked );
    connect( this->ui->btn_save, &QPushButton::clicked, this, &MainWindow::on_btn_save_clicked );
    connect( this->ui->btn_update, &QPushButton::clicked, this, &MainWindow::on_btn_update_clicked );

    file->open(QIODevice::ReadOnly | QIODevice::Text);
    QTextStream *readdst = new QTextStream( file );
    // QTextStream *readdst = new QTextStream( file, QIODevice::ReadOnly | QIODevice::Text );
    // readdst->setCodec( QTextCodec::codecForName( "GB2312" ) );
    allusers = new user();
    conn_users = new login();
    while( !readdst->atEnd() )           // 读取文件中保存的用户
    {
        user_num++;
        user *p = new user();
        p->account = readdst->readLine();
        p->psw = readdst->readLine();
        p->question = readdst->readLine();
        p->answer = readdst->readLine();
        secret(p->psw);
        secret(p->answer);
        p->next = allusers->next;
        p->inout = 0;
        allusers->next = p;
        qDebug()<<p->account<<p->psw<<p->question<<p->answer<<"\n";
    }
    file->close();
    // this->m_socket = new QTcpSocket( this );
    m_server = new QTcpServer( this );
    m_server->listen( QHostAddress::Any, 2001 );           // 服务器端口号：1997
    connect( this->m_server, SIGNAL( newConnection() ), this, SLOT( on_conned() ) );
    QTextCodec::setCodecForLocale( QTextCodec::codecForName( "GBK" ) );
}

MainWindow::~MainWindow() {
    delete ui;
}

void MainWindow::on_conned()
{
    login* p = new login(m_server);
    p->next = conn_users->next;
    conn_users->next = p;
    connect(p, SIGNAL( newMessage(login*) ), this, SLOT( on_readyread(login*) ) );
}

void MainWindow::on_disconned() {
}

void MainWindow::on_readyread(login*sender)
{
    QByteArray arr = sender->socket->readAll();
    QDataStream rdst(arr);
    QString head;
    rdst>>head;
    if( head == "reg" )           // 注册
    {
        rdst>>account>>psw>>question>>answer;
        QDataStream wdst(&arr, QIODevice::ReadWrite);
        wdst<<QString( "reg" );
        if( allusers->finduser(account) )           // 如果用户已存在，返回 fail
            wdst<<QString( "fail" );
        else
        {
            user_num++;

```




```

        user* p = new user();
        p->account = account;
        p->psw = psw;
        p->question = question;
        p->answer = answer;
        p->next = allusers->next;
        p->inout = 0;
        allusers->next = p;
        wdst<<QString("ack");
    }
    wdst<<account;
    sendmessage(arr, sender->socket);    // 之后应析构?
}
else if(head=="login")    // 登录
{
    QString ip;
    quint16 port;
    rdst>>ip>>port>>account>>psw;
    QDataStream wdst(&arr, QIODevice::ReadWrite);
    wdst<<QString("login");
    user* p =allusers->finduser(account);
    if(p )
    {
        if(p->psw==psw) // 账号密码均正确
        {
            client_num++;
            p->user_port =port;
            wdst<<QString("ack")<<account;
            p->ip = ip;
            p->inout = 1;
            sender->valid = 1;
            sender->inout = 1;
            sender->account = account;
            sender->ip = ip;
            if(p->offmsg)    // 用户存在离线消息
            {
                wdst<<QString("offmsg");
                readOffMsg(p->account, wdst);
                p->offmsg = 0;
            }
            else wdst<<QString("nooffmsg");
        }
        else wdst<<QString("fail")<<account;
    }
    else wdst<<QString("none")<<account;
    sendmessage(arr, sender->socket);
}
else if(head=="logout")    // 登出
{
    rdst>>account;
    user * p=allusers->finduser(account);
    if(p )
    {
        client_num--;
        p->inout = 0;
        sender->inout = 0;
    }
}
else if(head=="find")    // 找回密码
{
    rdst>>account>>question>>answer>>psw;
    QDataStream wdst(&arr, QIODevice::ReadWrite);
    wdst<<QString("find");
    user * p=allusers->finduser(account);
    if(p )
    {
        if(p->question==question)

```



```

        {
            if (p->answer==answer)
            {
                p->psw = psw;
                wdst<<QString("ack");
            }
            else
                wdst<<QString("fail");
        }
        else wdst<<QString("wrong");
    }
    else wdst<<QString("none");
    wdst<<account;
    sendmessage(arr, sender->socket);
}
else if (head=="getfriend")    // 获取用户列表
{
    rdst>>account;
    QDataStream wdst(&arr, QIODevice::ReadWrite);
    user* p = allusers->next;
    wdst<<QString("friend")<<user_num-1;    // 总用户数量
    while( p )
    {
        if (p->account != account)
            wdst<<p->account;
        p =p->next;
    }
    sendmessage(arr, sender->socket);
}
else if (head=="applyforip")    // 申请IP
{
    QString chatter;
    rdst>>account>>chatter;
    arr.clear();
    QDataStream wdst(&arr, QIODevice::ReadWrite);
    wdst<<QString("getip");
    user*p=allusers->finduser( chatter);
    if (p->inout)    // 在线
        wdst<<QString("friendonline")<<p->ip<<p->user_port;
    else
        wdst<<QString("friendoff");
    sendmessage(arr, sender->socket);
}
else if (head=="chatmsg")    // 服务器收到（离线）消息
{
    QString msg, Sender, receiver;
    rdst>>Sender>>receiver>>msg;
    user* p=allusers->finduser(receiver);
    p->offmsg = 1;

    qDebug()<<"offmsg:"<<p->offmsg;

    writeOffMsg(Sender, receiver, msg);
}
}

void MainWindow::on_btn_exit_clicked()
{
    on_btn_save_clicked();
    allusers->destroy();
    conn_users->destroy();
    this->close();
}

void MainWindow::sendmessage(QByteArray &arr, QTcpSocket *&client){
    client->write(arr);
}

void MainWindow::secret(QString &message)    // 密码加密函数
{

```

```
QTextCodec::setCodecForLocale(QTextCodec::codecForName("GBK"));
std::string str = message.toString();
int i = str.length();
char *s = (char*)str.c_str();
for(int j=0;j<i;++j)
    s[j] = s[j] ^ 0x04;
message = QString(s);
}

void MainWindow::writeOffMsg(QString &sender, QString &receiver, QString &msg)
{
    QString filename = receiver + QString(".txt");
    file=new QFile(filename);
    file->open(QIODevice::ReadWrite | QIODevice::Text);
    QTextStream writedst(file);
    writedst.seek(file->size());
    qDebug()<<file->size(); // 输出文件大小
    writedst<<sender<<"\n";
    writedst<<msg<<"\n";
    file->close();
}

void MainWindow::readOffMsg(QString &receiver, QDataStream &dst)
{
    QString filename = receiver + QString(".txt");
    file=new QFile(filename);
    file->open(QIODevice::ReadWrite | QIODevice::Text);
    QTextStream readdst(file);
    while(! readdst.atEnd())
    {
        dst<<(readdst.readLine());
        dst<<(readdst.readLine());
    }
    file->close();
    file->open(QFile::WriteOnly | QFile::Truncate); // 用来清空文件
    file->close();
    QFile::remove(filename);
}

void MainWindow::on_btn_save_clicked()
{
    file = new QFile("Register.txt");
    file->open(QFile::WriteOnly | QFile::Truncate); // 用来清空文件
    file->close();
    file->open(QIODevice::WriteOnly | QIODevice::Text);
    QTextStream writedst(file);
    user * p=allusers->next;
    for( ; p;p=p->next) // 保存用户信息
    {
        secret(p->psw);
        secret(p->answer);
        qDebug()<<p->account<<p->psw<<p->question<<p->answer;
        writedst<<p->account<<"\n";
        writedst<<p->psw<<"\n";
        writedst<<p->question<<"\n";
        writedst<<p->answer<<"\n";
        secret(p->psw);
        secret(p->answer);
    }
    file->close();
}

void MainWindow::on_btn_update_clicked() // 刷新用户信息
{
    ui->list_off->clear();
    ui->list_online->clear();
    user * p=allusers->next;
    for( ; p;p=p->next)
    {
        if(p->inout)
```

```
        ui->list_online->addItem(p->account+"_"+p->ip);
    else
        ui->list_off->addItem(p->account);
    }
}
```

A.2.2 用户类

```
#include "user.h"

user::user()
{
    next = 0;
    inout = 0;
    user_port = 0;
    offmsg = 0;
}

user::~user()
{
}

user* user::finduser(QString &account)
{
    user *p=this->next;
    while(p)
    {
        if(p->account==account)
            return p;
        p = p->next;
    }
    return 0;
}

void user::destroy()
{
    user*p = this->next, *pp=p;
    while(pp)
    {
        pp=p->next;
        free(p);
        p = pp;
    }
    this->next = 0;
}
```

A.2.3 登录

```
#include "login.h"

login::login(QTcpServer*server)
{
    next =0;
    inout =0;
    valid = 0;
    socket = new QTcpSocket(this);
    socket = server->nextPendingConnection();
    connect(socket,SIGNAL(readyRead()),this,SLOT(on_connected()));
    connect(socket,SIGNAL(disconnected()),this,SLOT(on_disconned()));
    qDebug()<<socket->peerName()<<socket->peerAddress()<<socket->peerPort();
}

login::login()
{
    next =0;
    inout =0;
    valid = 0;
    socket = 0;
}
```

```
void login:: destroy()
{
    login*p = this->next, *pp=p;
    while(pp)
    {
        pp=pp->next;
        delete p;
        p = pp;
    }
    this->next = 0;
}
void login:: on_connected()
{
    emit newMessage(this);
}
void login:: on_disconned()
{
}
login::~~login()
{
    delete socket;
}
```