P2P 聊天软件概要设计说明书

赵俊龙 201521010304 通信学院

引言

本文档的目的是阐述 P2P 聊天软件的概要设计。本概要设计说明书编写的目的在于全面说明 P2P 聊天软件的设计考虑，包括程序系统的基本数据结构等信息。

需求分析

需求规定

设计和实现 P2P 聊天软件，完成用户界面（GUI）设计，同时实现一个用户和多个用户同时进行聊天。目的在于设计 GUI、多线程（异步）和套接字编程。各个客户端使用 IP 地址和 TCP/IP 监听端口号进行标识。每个客户端提供一个名字，方便其他客户端。

运行环境

实现平台为 Linux 或 Windows，编程语言为 C/C++/JAVA。

系统逻辑设计

本系统采用 Qt 和 C++技术，并充分利用了 Qt 提供的信号—槽机制来实现异步处理，基于 Windows 的开发环境进行程序设计和实现。

本系统主要实现两方面的逻辑功能：成员注册管理和聊天通信。

成员注册管理

P2P 聊天中各个成员的管理。每个成员称为一个 Peer，要有一个中心的服务器处理各个 Peer 的等级以及对已经登记的 Peer 进行更新。

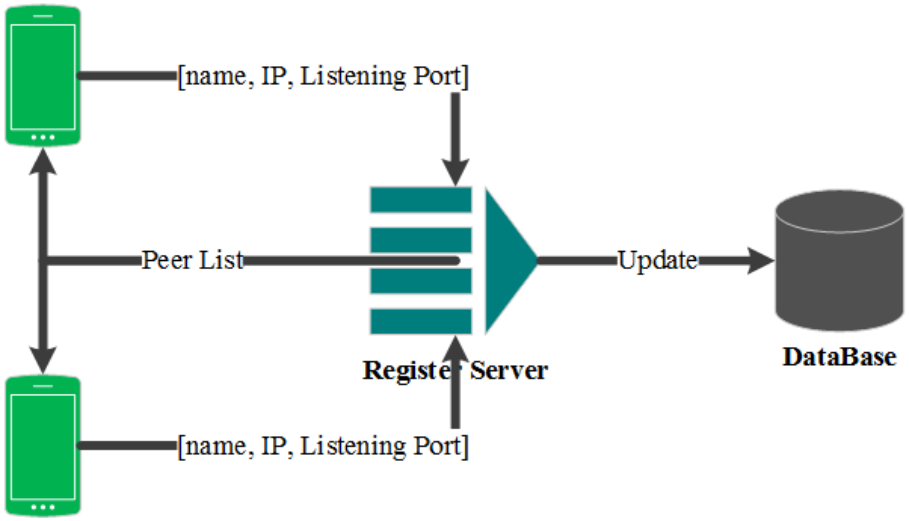
步骤一：一个 Peer 选择一个名字，通过登记界面进行登记。

步骤二：注册服务器拥有一个公有 IP，在一个众所周知的端口进行监听。Peer 向注册服务器 Register 请求，同时发送自己的登记信息（如自己的名字，自己的监听端口号等等）。注册服务器收到 Register 请求后，如果这是新的登记请求，服务器建立一个用于记录以及登记的 Peer 的列表（PeerList），否则更新 PeerList。

步骤三：服务器向来登记的 Peer 发送 PeerList。PeerList 由一些活跃的 Peer 组成，一个活跃的 Peer 是在最近 30 秒内进行登记的 Peer。

步骤四：每个已经登记的 Peer 每隔 15 秒向服务器登记，以使自己保持活跃，同时获取最新登记的 PeerList 信息。如果需要，Peer 更新自己维护的 PeerList 信息。

整个过程及交互如下：



聊天通信

聊天通信过程通过直接的 P2P 实现，实现了多个 Peer 间的相互发送消息过程。

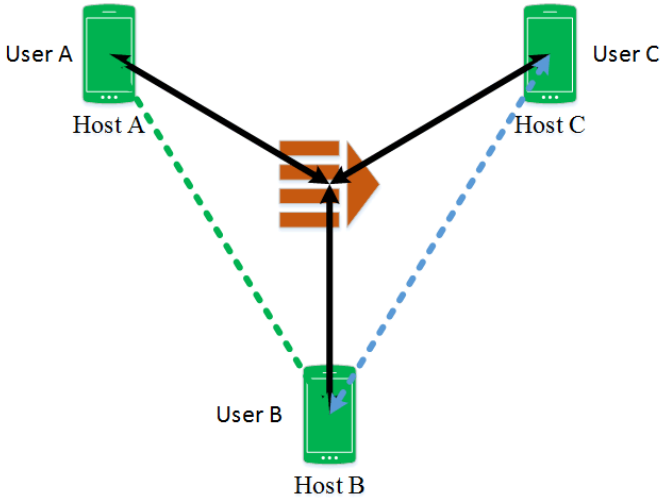
步骤一：在的客户端 P1 的用户 U1（U1@P1）在自己 Peer List 中选择一个 peer，假设为 U2@P2。

步骤二：P1 建立一个到 P2 的连接。

步骤三：U1 在自己的发送信息界面编写信息，然后点击发送按钮，P1 将信息通过步骤 2 中建立的连接发送到 P2。

步骤四：如果第 3 个用户 U3@P3 发送一条消息到 U1@P1，如果 U1 此时正好有一个信息界面与 U3 进行交互，那么就显示消息，如果没有，就要提示 U1 有新消息到来。

整个过程及交互如下：

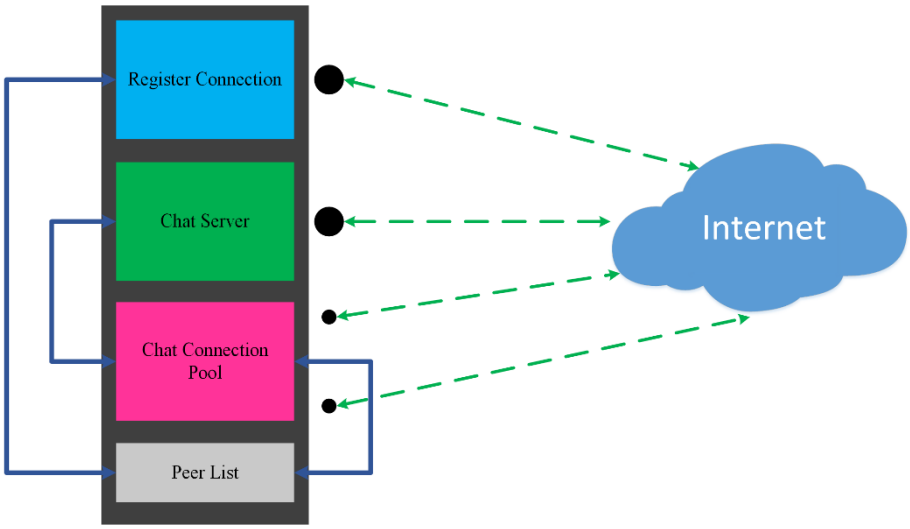


系统结构及接口设计

对于上面整个的过程和设计要求，我们分别设计了两个独立程序，分别是聊天客户端 ChatClient 和注册服务器 RegisterServer。下面我们就对 ChatClient 和 RegisterServer 的架构进行详细说明。

ChatClient 架构

ChatClient 架构如下图：



ChatClient 主要划分为四个部分，分别是 Register Connection、Chat Server、Chat Connection Pool 以及 PeerList，分别对应为注册连接管理、聊天通信服务器、聊天通信连接池以及 PeerList 数据结构。

关键数据结构

```
QList<PeerInfo *> peerList;   // 记录活跃的远端主机
QList<ChatConnection *> connList;   // 用于管理所有聊天连接
ChatServer *server;   // 本地聊天连接监听服务器
RegisterConnection *registConnection;   // 与中心服务器相连接
```

Register Connection

Register Connection 与远端注册服务器相挂接，执行注册操作，以及从远端注册服务中心服务器获取最新的 PeerList 信息，同时通过周期注册来保持激活状态。

具体设计如下：

核心数据结构

```
QTimer periodicTimer;
QString peerListString;
QString registerMessage;
```

periodicTimer 为一个周期定时器，该定时器定时触发发送注册信息的动作。

peerListString 为 Register Connection 从远端注册中心服务器获取到的 PeerList 信息字符串。ChatClient 可以通过解析该字符串来更新系统的 PeerList 信息。

registerMessage 为注册信息字符串，包含了本 ChatClient 的 ChatServer 的地址信息（IP 地址和端口号）以及标识名。

核心函数与接口

```
signals:
    void newPeerList(QString &peerlist);
slots:
    bool sendPeriodicRegisterMessage();
/***************工具函数*******************/
int readDataIntoBuffer(int maxSize = MaxBufferSize);
int dataLengthForCurrentDataType();
bool readProtocolHeader();
bool hasEnoughData();
void processData();
```

/***/

通过上述的工具函数，Register Connection 读取 socket 中接收到的信息，解析出接收到的数据，然后触发发送 newPeerList 的信号，通知系统有新的 PeerList 信息。

sendPeriodicRegisterMessage()函数由前面的数据结构 periodicTimer 定时器定时触发，即通过 register connection 向远端注册服务器进行再注册和激活操作。

小结

RegisterConnection 类结构主要完成与远端注册中心服务器的交互操作，包括建立连接，定时发送注册信息，获取新的 PeerList 信息。

ChatServer

ChatServer 为运行在 ChatClient 本地的聊天通信服务器，主要用于支持 P2P 聊天连接。

核心数据结构

```
QHostAddress myIp;
int myPort;
```

毫无疑问，这两个数据结构主要用于记录本地聊天服务器的 IP 地址以及监听端口号。

核心函数与接口

```
    bool startServer();
signals:
    void newChatConnection(ChatConnection *connection);
protected:
    void incomingConnection(qintptr socketDescriptor);
```

ChatServer 通过 startServer()函数启动 ChatServer 聊天服务器，同时通过 incommintConnection()函数监听和 Accept 远端的连接。当有新的连接时，创建新的 ChatConnection，然后触发发送 newChatConnection()信号，新的 ChatConnection 是该信号的参数。

小结

ChatServer 的主要作用就在于充当一个聊天服务器的作用，接收远端连接，通过信号的方式返回接收得到的远端 ChatConnection，并将其加入 connList 之中。

ChatConnection 连接池

ChatConnection 连接池，顾名思义，就是用于管理所有 ChatConnection 的实体。ChatConnection 类完成了一个聊天连接所有必需的数据操作和行为动作。该 ChatConnection 连接池将管理两种 ChatConnection，一种是由本 ChatClient 向远端聊天服务器主动发起会话连接的 Active 主动式 ChatConnection，另外一种是由远端的 ChatClient 向本地 ChatServer 发起会话连接请求的 Passive 被动式 ChatConnection。

核心数据结构

```
Int role;     // 主动连接或是被动连接？
QString userName;   // 通过远端的信息来标识该连接
QByteArray buffer;
```

其中，userName 用远端的 Peer 的标识名来表示；而 buffer 主要用于数据缓冲。

核心函数与接口

```
signals:
    void newMessage(QString from, QString message);
    void disconnectedChat(ChatConnection *);
    void socketError(ChatConnection *, QAbstractSocket::SocketError);
private slots:
    void handleDisconnected();
    void handleSocketError(QAbstractSocket::SocketError);
```

每个 ChatConnection 会维护三个信号，分别是 newMessage、disconnected 以及 socketError。newMessage 在接收到新消息时触发，同时通过参数传递 message；而 disconnected 以及 socketError 在连接断开或 socket 错误时触发。

另外，每个连接对于连接断开以及 socket 错误的情况也都有相应的处理过程。

小结

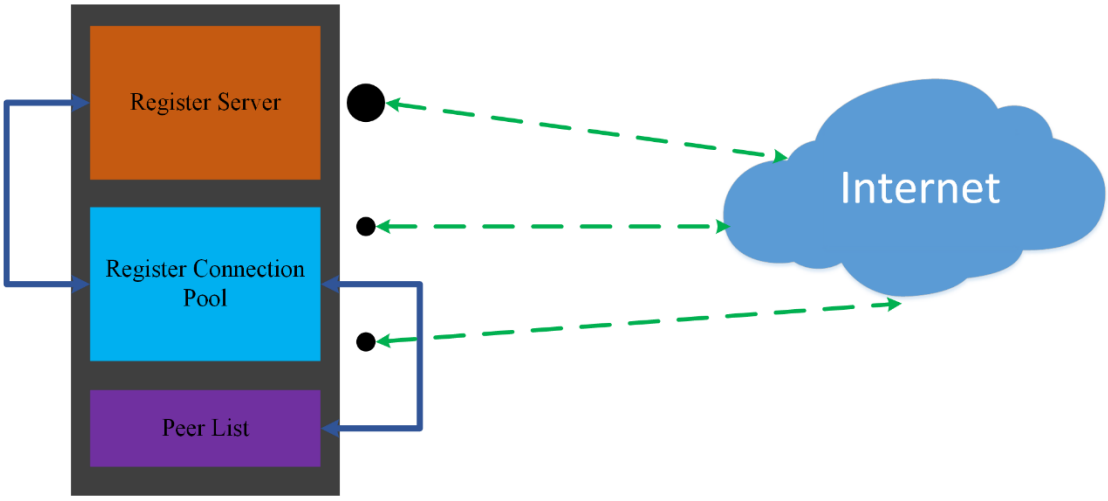
ChatConnection 实现了聊天通信连接的管理与控制，通过信号槽机制实现接受和发送信息，而具体的接收和发送信息的过程在 ChatDialog 的图形界面之中实现。

PeerList 管理

我们通过前面所讲的关键数据结构来管理整个系统的所有信息。另外，结合后面所讲的 GUI 界面的实现，就可以完成整体的信息管理与控制。

RegisterServe 架构

Register Server 的架构如下图所示：



同 ChatClient，RegisterServer 维护了三部分的功能模块，分别是 RegisterServer、RegisterConnection Pool 以及 PeerList Manager，分别对应于注册服务器例程、注册连接池以及 PeerList 信息管理模块。

关键数据结构

```
RegisterServer *server;     // 本地注册中心服务器
QList<RegisterConnection *> registerConns;   // 用于管理所有注册连接
QList<PeerInfo *> peerList; // 记录所有注册的主机信息
```

RegisterServer

Register Server 功能同 Chat Server，主要执行所有注册服务器例程，监听在本地 IP 和本地端口，然后在监听到新连接后执行 Accept 过程。

核心数据结构

```
QHostAddress myIP;
int myPort;
```

核心数据结构即是本地 IP 和监听端口，用于启动本地中心注册服务器。

核心函数与接口

```
    bool startServer();
signals:
    void newRegisterConnection(RegisterConnection *connection);
protected:
    void incomingConnection(qintptr socketDescriptor);
```

RegisterServer 通过 startServer()函数启动 RegisterServer 中心注册服务器，同时通过 incommintConnection()函数监听和 Accept 远端的连接。当有新的连接时，创建新的 RegisterConnection，然后触发发送 newRegisterConnection()信号，新的 RegisterConnection 是该信号的参数。

小结

RegisterServer 执行了中心注册服务器的监听和新建注册连接过程。

RegisterConnection 管理

RegisterConnection 管理由关键数据结构进行集中管理，而对于单独每个 RegisterConnection，其功能实现通过单独的 RegisterConnection 类来实现。

核心数据结构

```
QString registerMessage;
QByteArray buffer;
```

registerMessage 表示远端传递而来的注册信息，包括对端的 IP 地址、端口号

以及标识名。

buffer 表示发送和接收缓冲区，主要用于处理欲发送和接收到的信息。

核心函数与接口

signals:

void newRegisterMsg(RegisterConnection *, QString);

void registerError(RegisterConnection *, QAbstractSocket::SocketError);

public slots:

bool sendPeriodicPeerlistMessage(QString message);

void handleSocketError(QAbstractSocket::SocketError);

系统通过 newRegisterMsg 信号将该 socket 接收到的 register 注册信息发送至 GUI 程序，另外，系统会对每次的注册行为执行发送 sendPeriodicPeerlistMessage 过程，以通知远端主机上的 ChatClient。另外，系统会有处理 SocketError 的处理函数。

小结

RegisterConnection 实现了对单个注册连接行为的描述，而 RegisterConnection 连接池实现了所有注册连接的管理与控制。

PeerList 管理

我们通过前面所讲的关键数据结构来管理整个系统的所有信息。另外，结合后面所讲的 GUI 界面的实现，就可以完成整体的信息管理与控制。

系统界面设计

ChatClient

ChatClient 界面主要由三部分组成，分别是注册界面、主界面以及聊天界面。

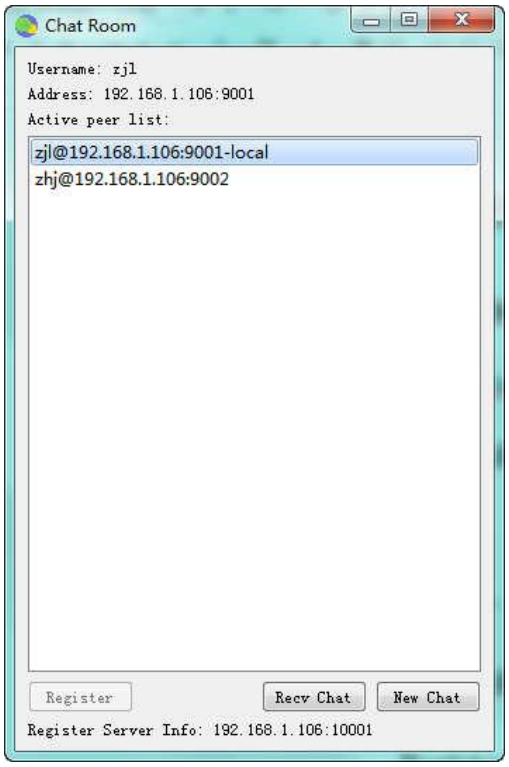
注册界面

界面代码：

```
QHBoxLayout *userLayout = new QHBoxLayout;
userLayout->addWidget(this->loginUsername);
userLayout->addWidget(this->editUsername);

QHBoxLayout *ipLayout = new QHBoxLayout;
ipLayout->addWidget(this->localIPAddress);
ipLayout->addWidget(this->localIPAddressBox);

QHBoxLayout *portLayout = new QHBoxLayout;
portLayout->addWidget(this->localPort);
portLayout->addWidget(this->localPortBox);

QHBoxLayout *ripLayout = new QHBoxLayout;
ripLayout->addWidget(this->registerIpLabel);
ripLayout->addWidget(this->registerIpEdit);

QHBoxLayout *rportLayout = new QHBoxLayout;
rportLayout->addWidget(this->registerPortLabel);
rportLayout->addWidget(this->registerPortBox);

QHBoxLayout *buttonLayout = new QHBoxLayout;
buttonLayout->addStretch();
buttonLayout->addWidget(this->doneButton);
buttonLayout->addStretch();
buttonLayout->addWidget(this->cancelButton);
buttonLayout->addStretch();
```

```
    QVBoxLayout *mainLayout = new QVBoxLayout;
    mainLayout->addWidget(this->loginLogo);
    mainLayout->addLayout(userLayout);
    mainLayout->addLayout(ipLayout);
    mainLayout->addLayout(portLayout);
    mainLayout->addLayout(ripLayout);
    mainLayout->addLayout(rportLayout);
    mainLayout->addStretch();
    mainLayout->addLayout(buttonLayout);


    this->setLayout(mainLayout);
```

主界面

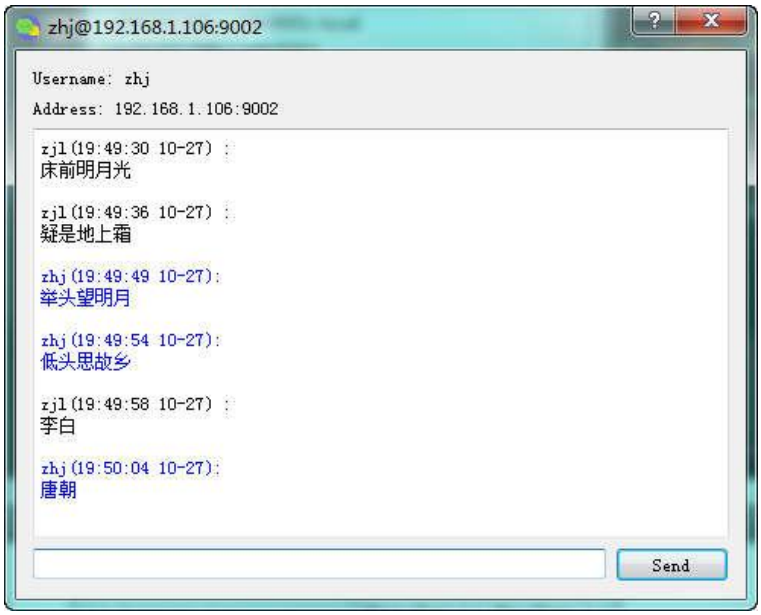


界面代码：

```
QVBoxLayout *myTextLayout = new QVBoxLayout;
    myTextLayout->addWidget(myName);
    myTextLayout->addWidget(myAddress);

    QHBoxLayout *chatLayout = new QHBoxLayout;
    chatLayout->addWidget(registerButton);
    chatLayout->addStretch();
    chatLayout->addWidget(checkButton);
```

```
    chatLayout->addWidget(chatButton);

    QVBoxLayout *mainLayout = new QVBoxLayout;
    //mainLayout->addLayout(myInfoLayout);
    mainLayout->addLayout(myTextLayout);
    mainLayout->addWidget(infoLabel);
    mainLayout->addWidget(peerListWidget);
    mainLayout->addLayout(chatLayout);
    mainLayout->addWidget(remoteServerLabel);

    QWidget *widget = new QWidget(this);
    widget->setLayout(mainLayout);

    this->setCentralWidget(widget);
```

聊天界面



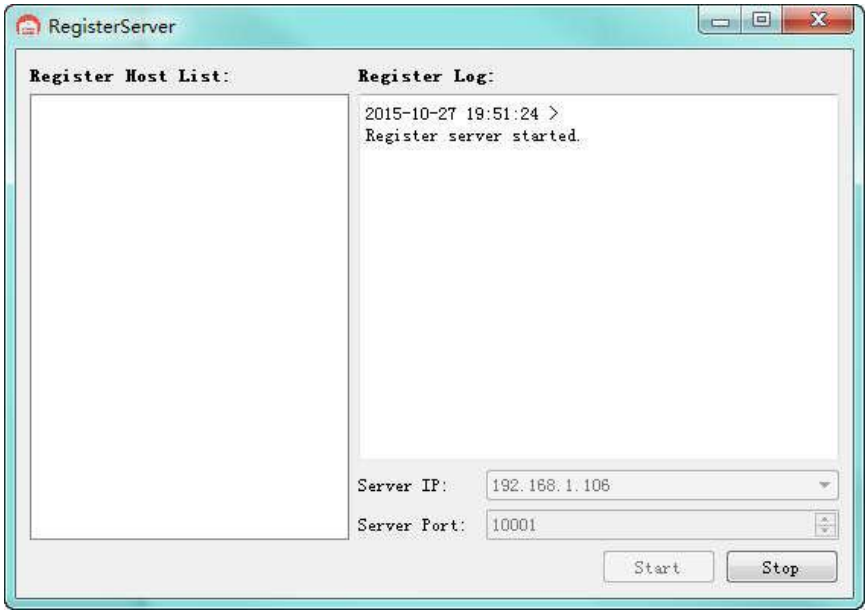
界面代码：

```
QVBoxLayout *infoLayout = new QVBoxLayout;
    infoLayout->addWidget(peerName);
    infoLayout->addWidget(peerAddress);

    QHBoxLayout *sendLayout = new QHBoxLayout;
    sendLayout->addWidget(msgSend);
    sendLayout->addWidget(sendButton);
    sendLayout->setStretch(0, 5);
    sendLayout->setStretch(1, 1);
```

```
QVBoxLayout *dialogLayout = new QVBoxLayout;
//dialogLayout->addLayout(peerLayout);
dialogLayout->addLayout(infoLayout);
dialogLayout->addWidget(logChat);
dialogLayout->addLayout(sendLayout);

this->setLayout(dialogLayout);
```

RegisterServer

主界面



界面代码：

```
QVBoxLayout *listLayout = new QVBoxLayout;
listLayout->addWidget(this->listLabel);
listLayout->addWidget(this->registerListWidget);

QHBoxLayout *ipLayout = new QHBoxLayout;
ipLayout->addWidget(this->ipLabel);
ipLayout->addWidget(this->ipBox);

QHBoxLayout *portLayout = new QHBoxLayout;
portLayout->addWidget(this->portLabel);
portLayout->addWidget(this->portBox);
```

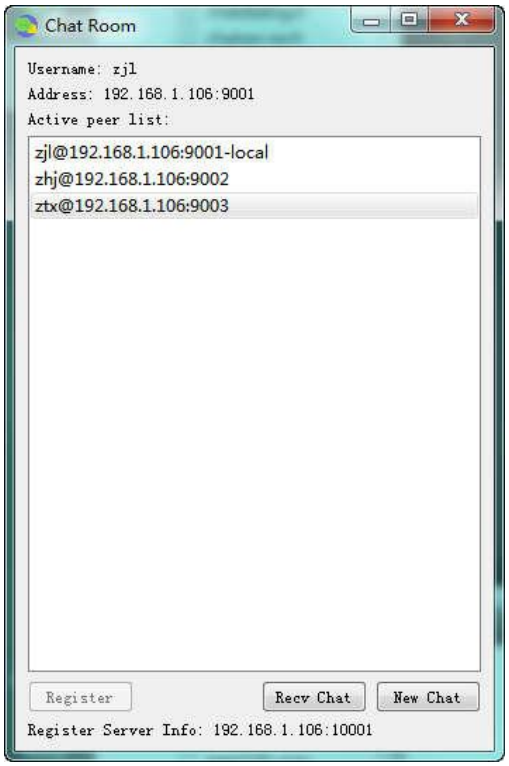
```
QVBoxLayout *logLayout = new QVBoxLayout;
logLayout->addWidget(this->logLabel);
logLayout->addWidget(this->logRegister);
logLayout->addLayout(ipLayout);
logLayout->addLayout(portLayout);

QHBoxLayout *buttonLayout = new QHBoxLayout;
buttonLayout->addStretch();
buttonLayout->addWidget(this->startButton);
buttonLayout->addWidget(this->stopButton);

QHBoxLayout *hLayout = new QHBoxLayout;
hLayout->addLayout(listLayout);
hLayout->addLayout(logLayout);

QVBoxLayout *mainLayout = new QVBoxLayout;
mainLayout->addLayout(hLayout);
mainLayout->addLayout(buttonLayout);

QWidget *widget = new QWidget(this);
widget->setLayout(mainLayout);


this->setCentralWidget(widget);
```

系统运行设计

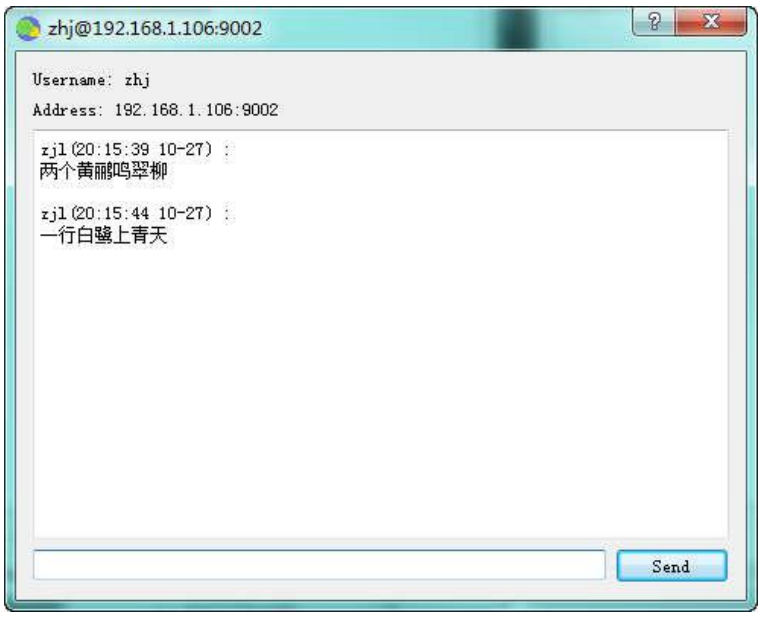
用户 1 操作

1、注册用户

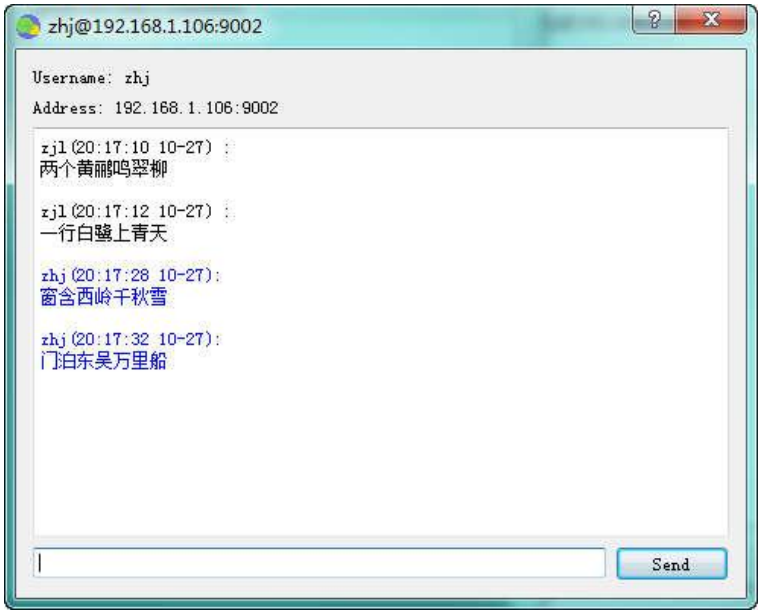
2、获取最新 PeerList



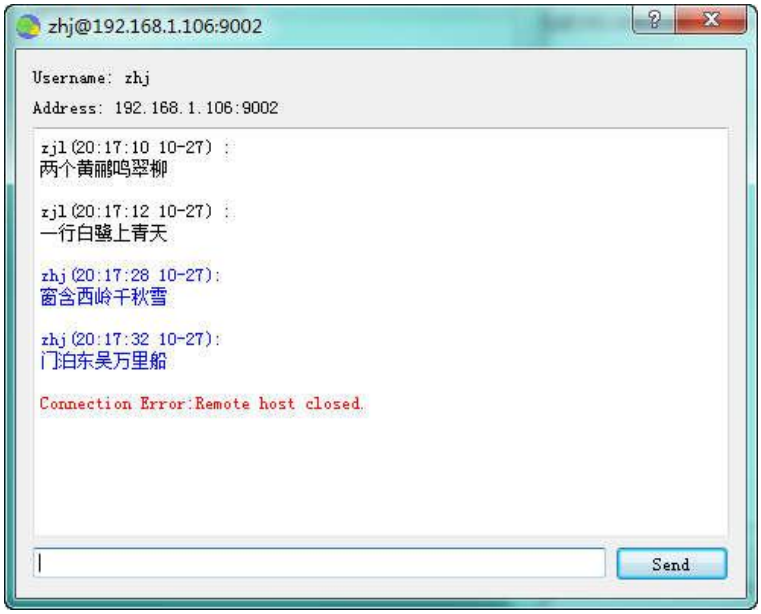
3、选取通信对象用户（假设为用户 2），点击 New Chat 按钮



4、发送消息



5、远端连接断开

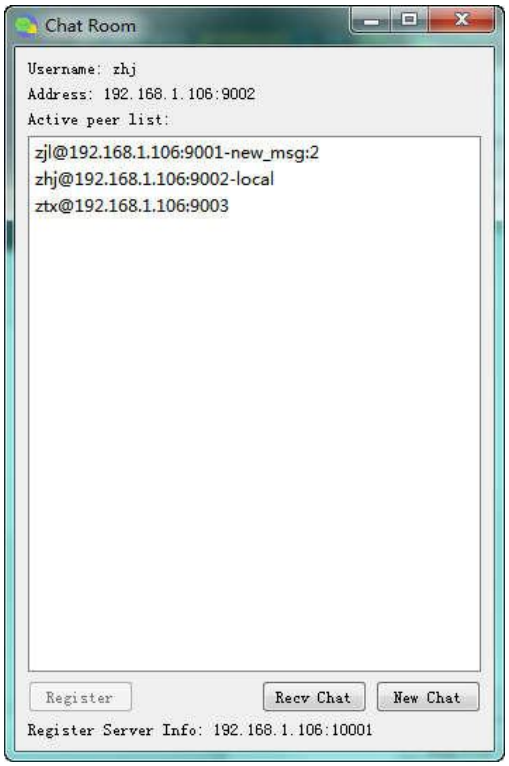


用户 2 操作

1、注册用户

2、获取最新 PeerList

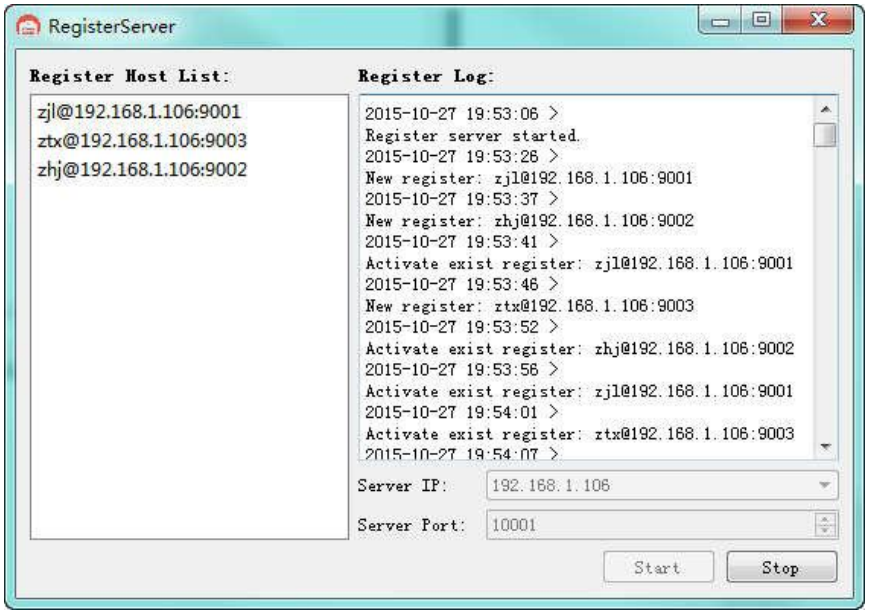
3、刷新接收消息



4、点击 Recv Chat 按钮接收消息，同时进行对话

中心注册服务器

主界面变化：



存在周期更新信息。

总结

在本系统中，本人实现了需求中的全部内容，实现了周期注册中心服务器，本地到远端的 P2P 聊天过程。鉴于个人水平有限，实现过程使用了异步操作，同时界面设计极其简陋，交互设计也不尽人意，望指正批评。