

基于Java Socket的聊天软件

1.1 环境

1.1.1 开发平台

系统版本：windows10 家庭中文版；处理器：Inter(R) Core(M) i5-4210U CPU @ 1.70GHz 2.40GHz；内存：8.00 GB(7.89 GB 可用)；系统类型：64 位操作系统。

开发平台：Eclipse Oxygen Release (4.7.0)，JDK 1.8.0_144，JavaFX Scene Builder 2.0，MySQL5.6.36-log，Navicat Premium11.2.7(64bit)。

第三方组件：mysql-connector-java-5.1.38.jar，sun.misc.BASE64Decoder.jar，sun-jndi-ldap.jar，jfoenix.jar。

1.1.2 运行平台

系统版本：windows10 家庭中文版；处理器：Inter(R) Core(M) i5-4210U CPU @ 1.70GHz 2.40GHz；内存：8.00 GB(7.89 GB 可用)；系统类型：64 位操作系统。

软件组件：Eclipse Oxygen Release (4.7.0)，JDK 1.8.0_144，MySQL5.6.36-log。

第三方组件：mysql-connector-java-5.1.38.jar，sun.misc.BASE64Decoder.jar，sun-jndi-ldap.jar，jfoenix.jar

1.2 系统功能需求

基于 TCP 和 UDP 协议实现一个即时通讯工具，具体功能要求包括：

- (1) 工具包括服务器端和客户端；
- (2) 具备用户注册、登录、找回密码功能（基于 TCP 协议）；
- (3) 两个用户如果同时在线，采用点到点通信方式进行聊天，信息不需要通过服务器中转，服务器也不保存（基于 TCP 协议）；

-
- (4) 支持离线消息（基于 TCP 协议）；
 - (5) 支持点到点可靠文件传输（基于 UDP 协议）；
 - (6) 存储在服务器端的数据需要进行强加密；
 - (7) 支持不少于两组用户同时在线交流和传输文件；
 - (8) 文件传输具有良好的性能，能够充分利用网路带宽；
 - (9) 人机交互友好，软件易用性强。
 - (10) 客户端和服务端具备完成所需功能的基本图形用户界面（GUI），并具友好性。

1.3 系统设计

1.3.1 功能模块划分

系统分为客户端模块、服务器模块、文件传输模块、数据库访问模块、客户端 UI、加密模块等模块。其中客户端模块包括与其他客户端实现点对点聊天和与服务器端的通信来实现注册、登陆、修改密码、获取用户信息、发送离线消息等功能；服务器模块主要是监视客户端连接请求与客户端实现通信，以及与数据库进行交互获取相关信息并反馈给客户端以及简单的 UI；文件传输主要包括接收和发送，实现文件传输；数据库访问模块主要是和数据库建立连接，访问数据库实现增、删、查、改功能；客户端 UI 主要通过图形界面与用户实现交互；加密模块采用 DEC 加密方法对字符串进行加密。

1.3.2 系统架构

1. 客户端调用关系图如图 1.1 所示，启动时只启动了客户端 UI，此时根据用户点击做出响应，并且启动客户端连接服务器线程。如果用户点击登陆，则读取用户输入，发送给服务器端，判断其用户密码是否正确，正确则跳转至聊天界面；如果用户点击注册，则跳转至注册界面，等待用户输入，点击注册后向服务器发送注册信息，判断其注册成功后返回登陆界面；如果用户点击修改密码，则跳转至修改密码界面，等待用户输入，点击修改后向服务器发送修改信息，验证密保问题是否正确，正确则修改密码，否则则不修改。客户端连接服务器后开始监测向服务器发送消息队列，如果队列非空，则提取出消息并发送，如果有消息从服务器发送，则将其放入到接收服务器队列。如

果用户点击其他用户聊天，则启动接收和发送消息线程，发送消息线程监测向其他用户发送消息队列，如果队列非空，则将消息取出，判断是否是自己需要发送的消息，是则发送，都在放回；接收消息接收到消息后将其放入 到接收其他用户消息队列；聊天界面将获取的消息放入向其他客户端发送消息队列。

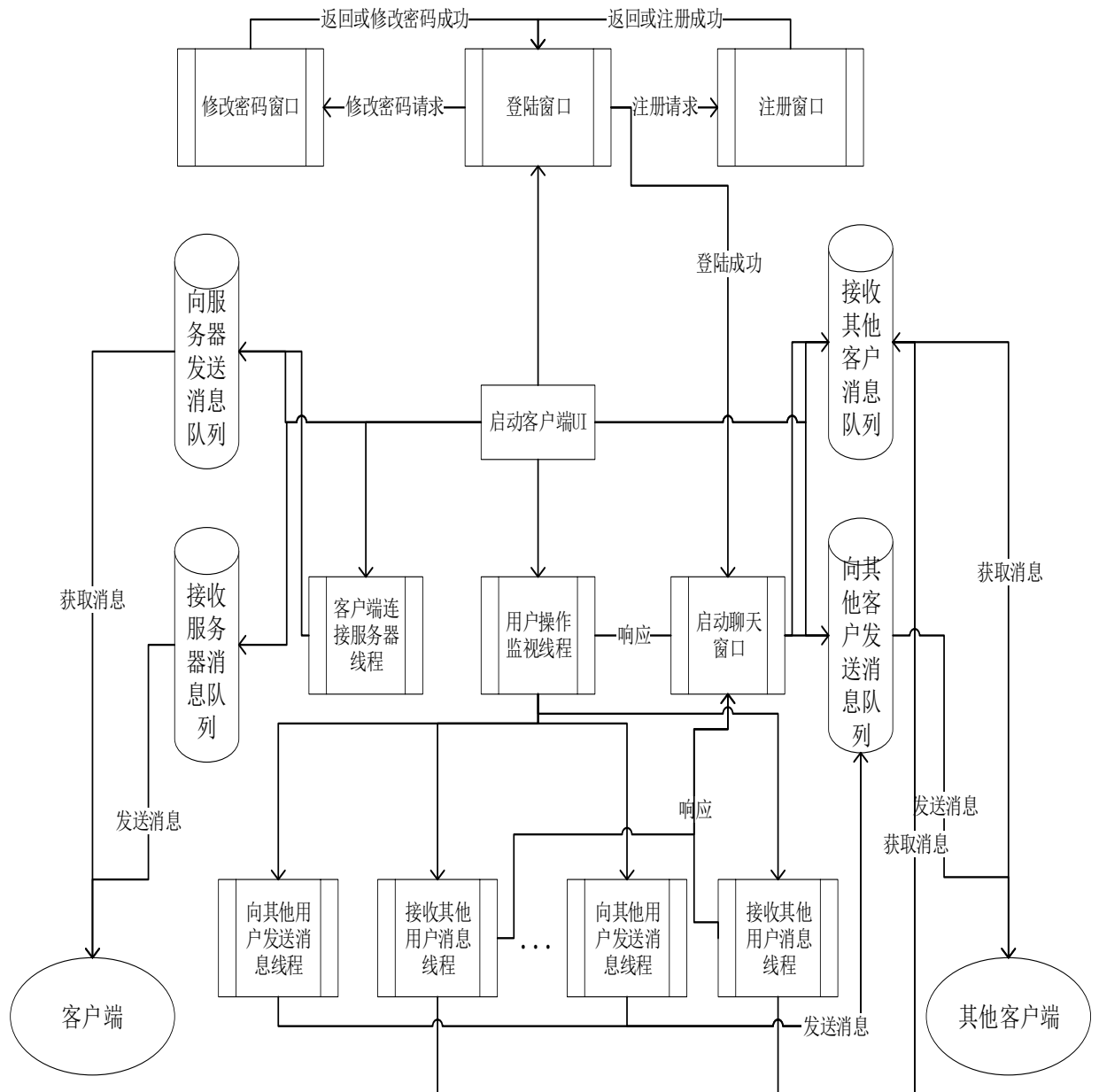


图 1.1 客户端调用关系图

2. 服务器端调用关系图如图 1.2 所示。服务器端监视线程一直运行，一旦发现请求连接，那么开启一个线程，与请求建立连接的用户建立连接，开始监视。一旦有消息接收，则根据消息类型与数据库进行交互，并且进行判断，然后反馈给用户，如果有需要储存的信息，则调

用加密算法对字符串进行加密，然后储存到数据库当中，需要从数据库读出信息时，调用解密算法对其进行解密，然后返回给处理消息线程。对于服务器界面，只是与数据库建立连接，然后调用加密解密算法进行显示即可。

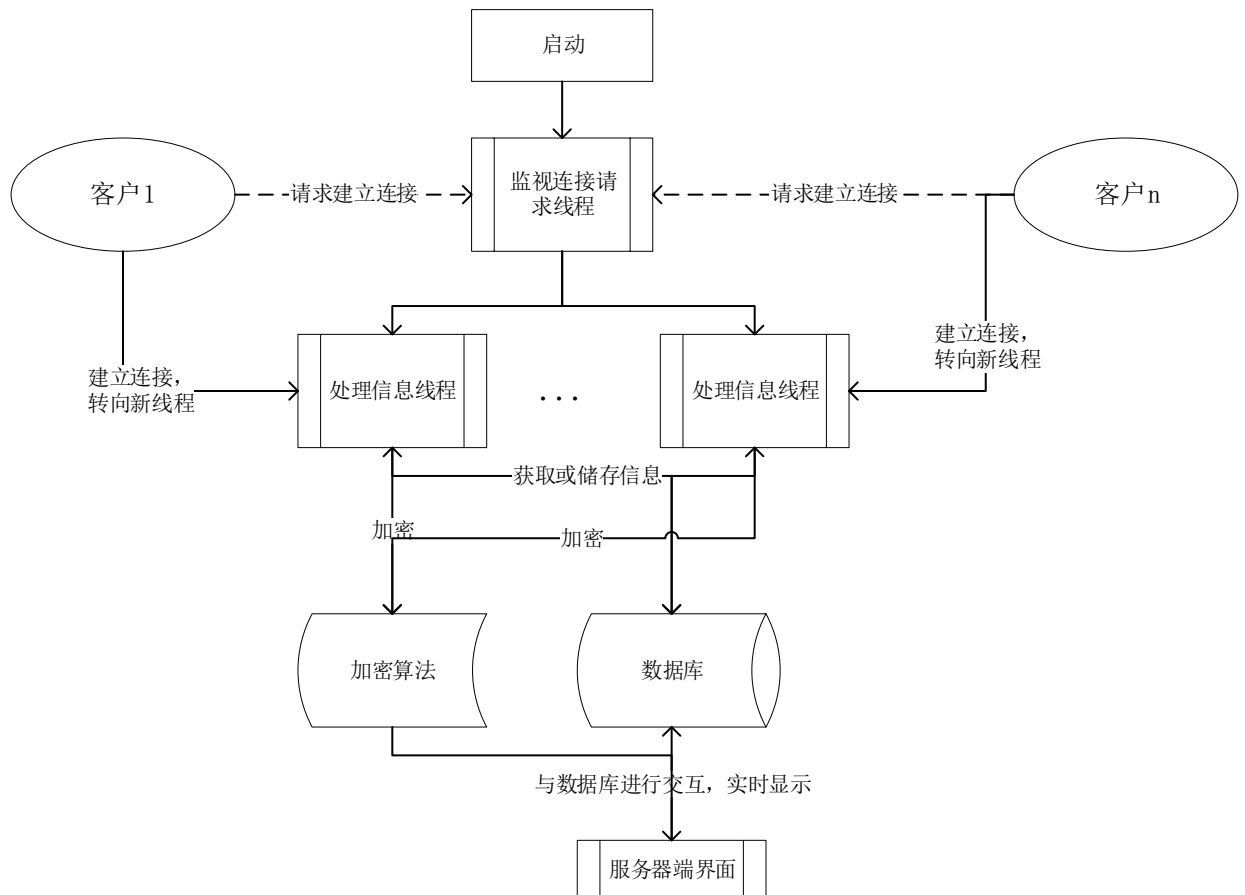


图 1.2 服务器端关系调用图

3. 使用 UDP 实现可靠文件传输，即传输过程中链路不可靠，可采用课本上有限状态机模型，如图 1.3、1.4 所示，进行文件传输。

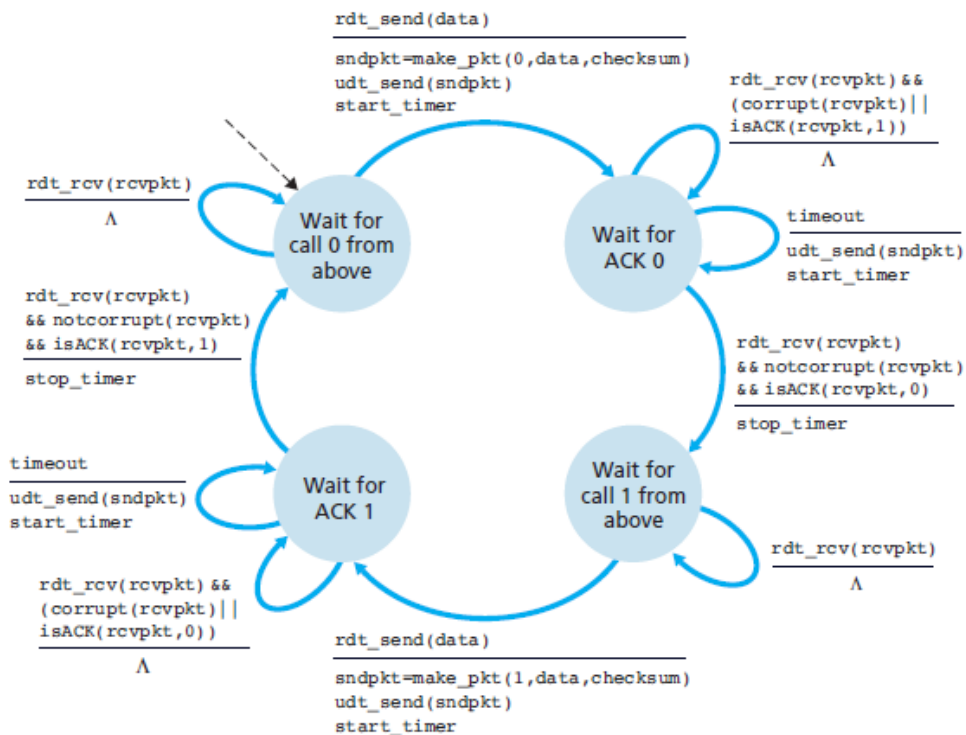


图 1.3 文件传输发送方状态机

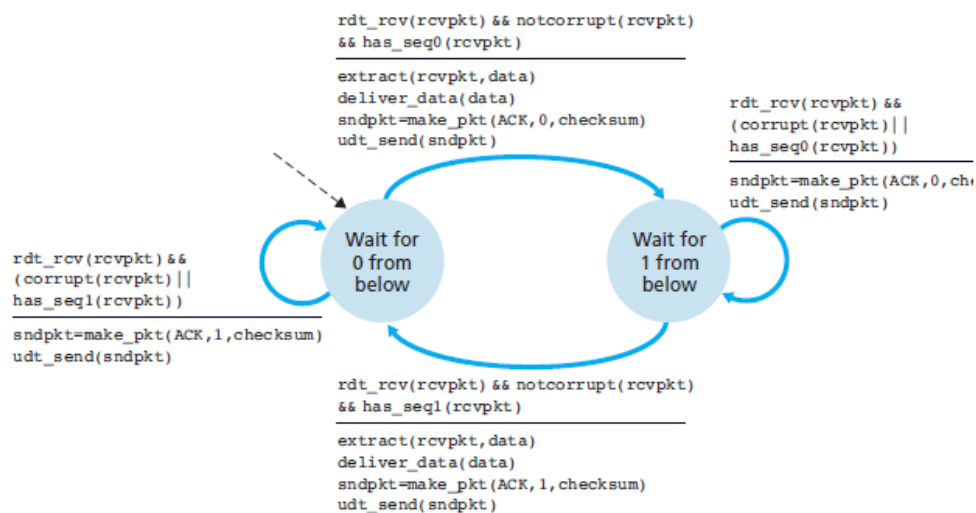


图 1.4 文件传输接收方状态机

1.3.3 应用层协议设计

- 1) 本系统用到了 TCP 和 UDP 协议，采用 JAVA 语言进行设计。TCP 使用其标准库中的 Socket 和 ServerSocket 类的相关方法来进行实现。UDP 使用标准库中的 DatagramPacket 和 DatagramSocket 来进行实

现。实现 UDP 的可靠文件传输，基于其标准库的 UDP 协议，采用拥塞窗口、ACK 回复、GBN、快速重传和超时重传技术实现，具体见系统实现。

- 2) 向服务器发送消息，主要分为注册信息，登陆信息，修改密码，查询端口，获取其他用户信息，发送离线消息，下线等，具体类型如表 1.1 所示。

表 1.1 向服务器发送消息字符串类型

消息类型		对应编号	消息内容
注册	用户名	00	“00”+用户名
	密码	01	“01”+密码
	密保问题	02	“02”+密保问题
	密保答案	03	“03”+密保答案
登陆	用户名	10	“10”+用户名
	密码	11	“11”+密码
修改 密码	用户名	20	“20”+用户名
	密保答案	21	“21”+密保答案
	新密码	22	“22”+新密码
离线消息		50	“50”+离线消息
查询端口		60	“60”+用户名
下线		70	“70end”

- 3) 服务器向用户发送消息，主要分为登陆认证消息，登陆信息，修改密码信息，查询端口信息，用户信息，具体如表 1.2 所示。

表 1.2 服务器向客户端发送消息类型

消息类型		消息内容
注册	成功	"sign up successfully!"
	失败	"sign up unsuccessfully!"

登陆	成功	"sign in successfully!"
	失败	"sign in unsuccessfully!"
修改 密码	成功	"Change the password successfully!"
	失败	"Incorrect answer!"
查询	失败	"Not find the user!"
端口	成功	"@12"+端口, "@12"+IP
用户 信息	在线	"00"+在线用户信息
	离线	"01"+离线用户信息

1.4 系统实现

1.4.1 服务器端实现

1) 监视线程实现

声明一个 `serversocket` 对象，实现一个循环，调用 `serversocket` 的 `accept` 方法监视用户请求建立连接。由于 `accept` 方法为阻塞的，所以对于循环来讲不会一直占用 CPU，只会在接收到用户请求时才会进行下面的操作。一旦接收到请求建立，则新启动一个线程处理与该进程的通信，其对应的流程图如图 1.5 所示。

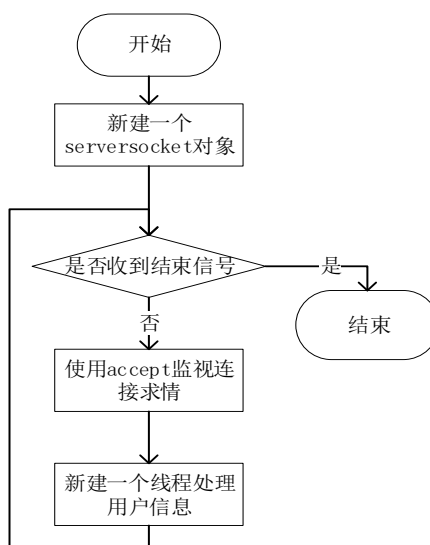


图 1.5 监视线程其对应的流程图

- 2) 用户消息处理线程实现。将用户发来的消息进行解析，主要获得之前表中列出的编号，进而判断其类型信息，然后使用 `case` 语句进行判断，调用加密解密算法并与数据库进行交互，实现相关判断。例如，对于注册消息，首先调用加密函数，对字符串进行加密，然后调用数据库类中的 `insert` 方法，如果返回为真，说明插入成功，即对应数据库中不存在该信息，也就是说还没有同名的用户名，那么向用户发送 "sign up successfully!"，如果插入失败，即对应数据库中已经有该信息，注册失败，向用户发送 "sign up unsuccessfully!"。其他操作类似该操作，其对应的流程图如图 1.6 所示。

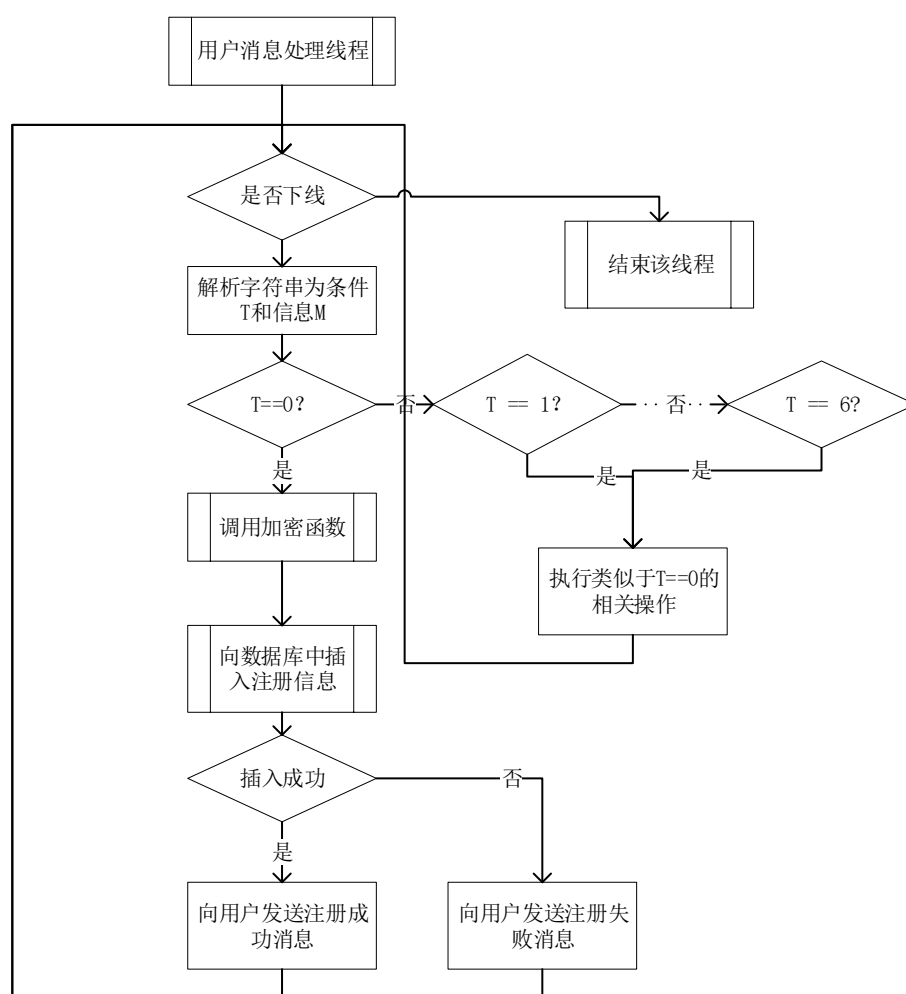


图 1.6 用户消息处理其对应的流程图

1.4.2 客户端实现

- 1) 一个客户端在启动后首先会随机分配一个未使用的端口，并且获取本机 IP 地址，然后启动 `ConnectionToServer`，与服务器建立连接，将其

端口和 IP 地址发送给服务器，以便其他用户在与其通信时能够通过服务器获取其 IP 地址和端口号，然后根据其发送给服务器的消息队列与服务器进行交互；ClientMonitor，监视其他用户向自己发送的建立连接请求；当需要与其他用户通信时，调用 startConClient 方法，该方法首先判断是否与对应用户建立连接，如果没有则实例化一个 ConnectionToClient 对象，与对应用户建立连接，并调用该对象的 start 方法，启动 Send 和 Receive 线程。调用关系图如图 1.7 所示。

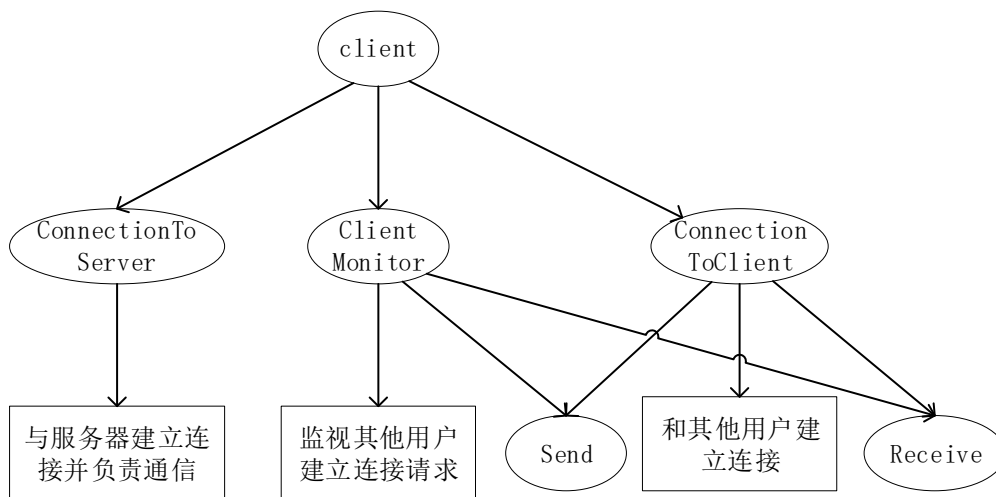


图 1.7 客户端调用关系图

- 2) Client 类对应一个用户，包括了图 1.7 所示的内容，主要是调用下层类的相关方法，主要方法如下：
 - a) startConServer 方法，启动 ConnectionToServer 和 ClientMonitor 两个线程，并将与其他客户发送和接收消息队列、与服务器发送和接收消息队列、当前连接客户信息链表、结束标志、socket 对象以及相关 UI 句柄等信息传递给两个线程。
 - b) startConClient 方法，判断是否与对应用户建立连接，如果已经建立连接，则不做任何操作，否则实例化 ConnectionToClient 对象，并调用其 start 方法，启动接收和发送消息进程，同时将该用户信息储存。
 - c) setExited 方法，将结束标志设置为 true，通知其他线程已经结束，需要退出。
- 3) ConnectionToServer 类在启动时会向服务器发送本机的 IP 地址和端口号，之后主要是等待服务器发送消息队列中的消息，一旦队列非空取出该消息，将其发送给服务器，并且根据发送消息的类型等待接收服

务器发送的回复消息，然后解析接收到的消息队列中，如果是相关命令，则将其加入到 task 任务队列当中，以供 GUI 模块中所述的 changeWindow 线程做出响应，如果是离线消息则将其加入到对应的消息队列当中，以便于用户切换到与该用户的聊天窗口时能够及时显示该用户的离线消息。判断发送消息类型的目的是，Java 的 socket accept 方法实现为阻塞赋值，必须收到消息后才能执行后面的程序，这样需要提前知道服务器会反馈几条消息，然后根据消息数量调用 accept 方法。

- 4) ClientMonitor 类主要是监视其他用户的建立连接请求，一旦发现其他用户请求建立连接，则新建 send 和 receive 线程，并将用户信息传递给对应线程，实现与对应用户的通信。由于监视连接请求的 accept 方法是阻塞的，所以这里直接实现成为了循环，其对应的流程图如图 1.8 所示。

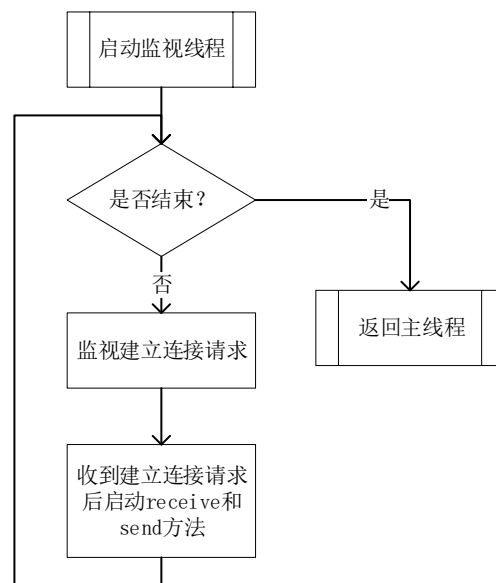


图 1.8 客户端监视连接请求

- 5) 一个 ConnectionToClient 类对应了一个点到点的连接，当用户需要与其他用户建立连接进行通信时，实例化 ConnectionToClient 对象，然后通过 start 方法新建 receive 和 send 线程，并传递其对应的收发消息用户名，以便于后续进行判断是否是该线程负责的消息，分别负责收发消息。
- 6) Send 类主要负责发送消息，其一旦启动便开始监视向其他用户发送消息队列，如果队列非空，则将该消息取出，首先判断是否是结束标

志，如果是，结束该线程，并将该标志放回到队列当中，如果不是则判断该消息是否是其负责的用户消息，如果是，则将其发送出去，如果不是，则将其放回到消息队列，以便于其他 send 线程可以获取到该消息，完成发送。其对应的流程图如图 1.9 所示。

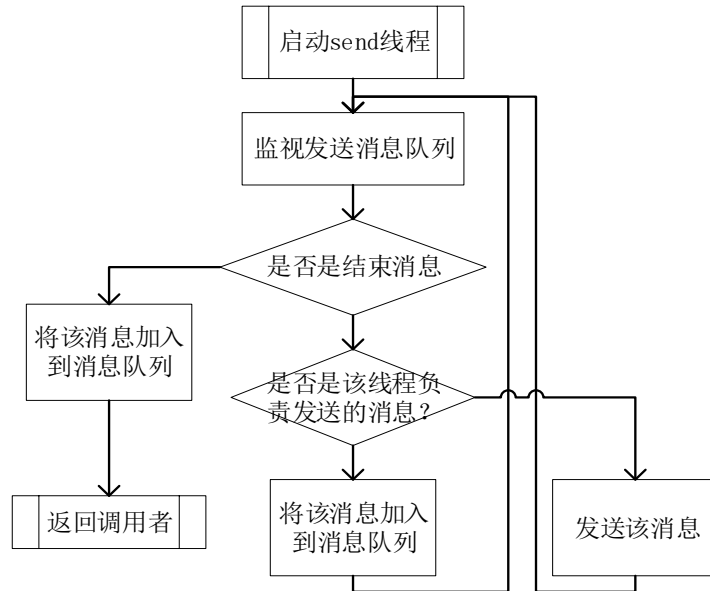


图 1.9 发送消息线程其对应的流程图

- 7) **Receive** 类主要负责接收消息，其一旦启动便开始使用阻塞方法 **receive** 来接收消息，一旦接收到消息，首先判断是否是断开连接请求，如果是则告诉父进程，然后结束程序，如果不是则判断当前显示的界面是否是发送消息方的聊天界面，如果是则直接在界面上显示该用户，如果不是，则需要将其加入到显示消息队列当中，那么首先判断队列中是否已经存在该用户的消息，如果存在，直接加入即可，否则新建一个 **list**，将该消息加入到 **list**，然后将该 **list** 加入到队列当中。其对应的流程图如图 1.10 所示。

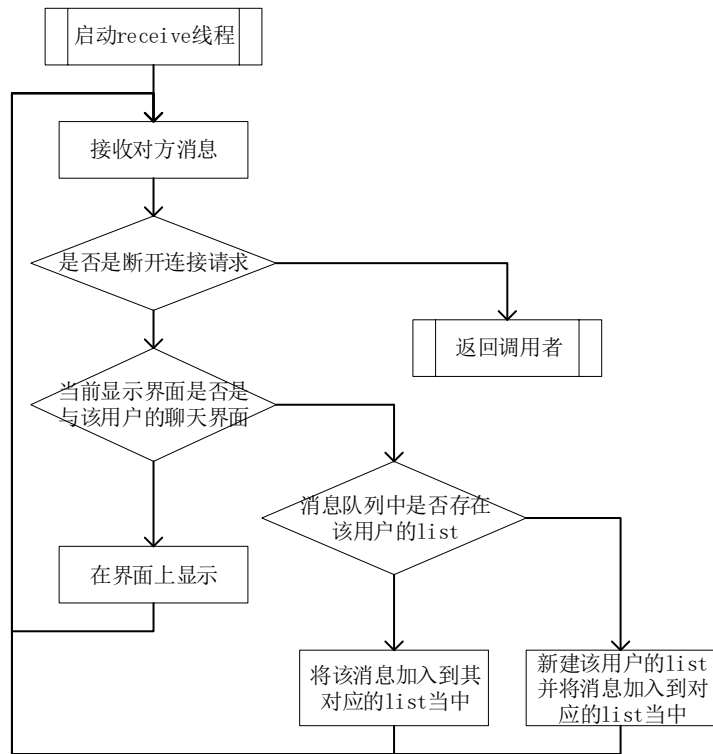


图 1.10 接收消息线程其对应的流程图

1.4.3 传输文件实现

- 1) 传输文件主要包括两个类，分别为接收文件类和发送文件类，通过 UDP 协议进行发送，采用 ACK 来判断包是否被接收方接收，采用滑动窗口、三次冗余 ACK 重传和超时重传技术实现可靠文件传输。
- 2) 发送文件类主要是通过设置窗口、等待 ACK 回复、计时等方法实现 UDP 的可靠文件传输，其对应的流程图如图 1.11 所示，主要方法如下。其中 window 采用链表实现，每次添加在链表尾部，删除在链表首部，模拟窗口的动态滑动。
 - a) setSequenceNumber 方法，在读取的对应字节首部加上该包对应的编号。
 - b) initWindow 方法，记窗口大小位 windowSize，首先读取 windowSize*1400 字节的文件，调用 setSequenceNumber 方法对其加上对应的编号进行封装并加入到 window 当中。
 - c) sendPacket 方法，按照顺序发送窗口种所有的包。
 - d) exmaineACK 方法，检测接收到的 ACK 与当前需要获得的 ACK 是否相等，相等返回 true，否则返回 false。

- e) `moveWindow` 方法，首先删除窗口中的第一个包的信息，然后判断是否已经读到文件尾部，如果读到，则不做任何操作，没有读到，读取 1400 字节，将其加入到窗口尾部，并使用 UDP 将其发送给接收方。
- f) `send` 方法，参数为接收方的 IP 地址，端口号和文件路径。首先打开文件，调用 `initWindow` 方法初始化窗口，调用 `sendPacket` 方法发送窗口中所有的包，然后等待 ACK，如果收到的 ACK 为当前所等待的 ACK 那么调用 `moveWindow` 方法，否则将 `wrongACK` 加 1，如果 `wrongACK` 为 4，则调用 `sendPacket` 发送窗口中所有的包，如果超时，同样调用 `sendPacket` 发送窗口中所有的包。

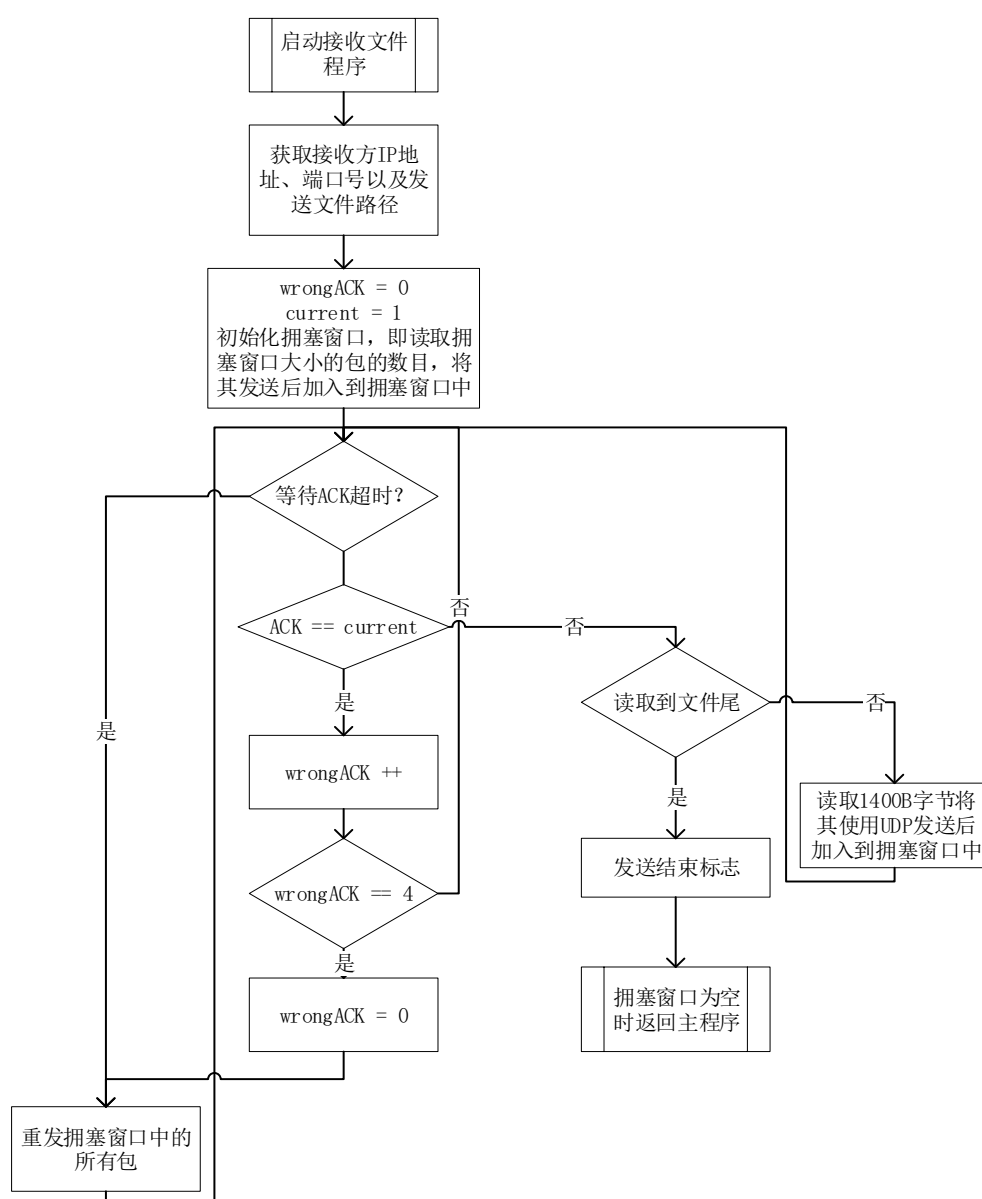


图 1.11 发送文件其对应的流程图

- 3) 接收文件类主要接收 UDP 包，向发送方发送 ACK、将文件写入到磁盘等方法实现文件接收，其对应的流程图如图 1.12 所示，主要方法如下。
- a) `getSequenceNumber` 方法，解析收到的包，获取其存储的 `sequence number`。
 - b) `getData` 方法，解析收到的包，获取其发送的去除首部之外的二进制信息。
 - c) `getACK` 方法，根据收到的 `sequence number` 来生成对应的 ACK。
 - d) `isend` 方法，根据收到的包的信息判断该包是否是结束包，如果是则返回 `true`，否则返回 `false`。
 - e) `receive` 方法，参数为发送方的 IP 地址、端口号以及文件名。首先新建该文件，然后使用 `DatagramSocket` 中的 `receive` 方法接收收到的包，收到包后通过 `getSequenceNumber` 方法解析收到的是不是正在等待的包，如果是则通过 `isend` 方法判断是否以及发送完成，完成则关闭文件结束程序，否则通过 `getData` 获取二进制信息并将其写入到磁盘，然后通过 `getACK` 方法生成该包的 ACK 发送给用户。

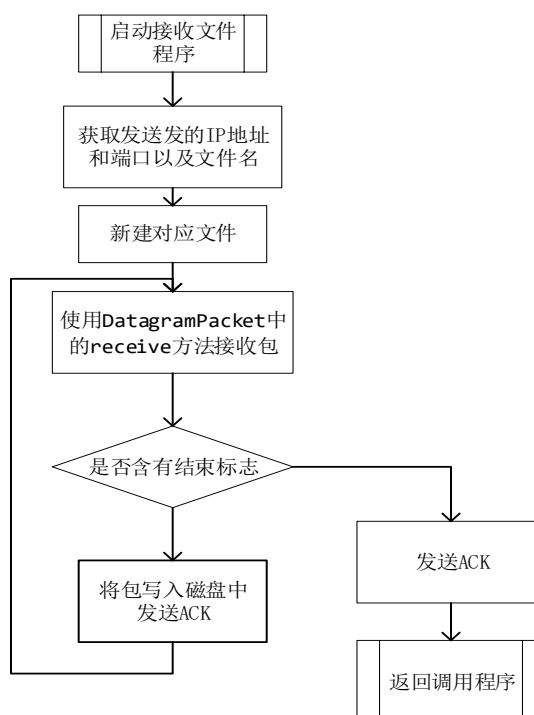


图 1.12 接收文件其对应的流程图

1.4.4 GUI 实现

- 1) GUI 主要使用 JvavaFX 实现，基本框架参考 <https://github.com/DomHeal/JavaFX-Chat>，并在其基础上做相关修改，实现本系统的 GUI。
- 2) 登陆、注册、修改密码框架相同，以登陆界面为例说明如何实现。在 JavaFX 中画出需要的图形，生成 FXML 文件，效果图如图 1.13 所示。加入 `fx:controller="wxs.org.login.LoginWindow"` 属性，并在对应按钮属性中加入 `onMousePressed` 属性，然后在对应路径中新建 `LoginWindow.java` 文件，构造对应方法。
 - a) `start` 方法，初始化相关参数。
 - b) `startConnectionToServer` 方法，判断是否已经和服务器建立连接，如果没有，则请求建立连接。
 - c) `checkKV` 方法，判断输入是否合法，如果不合法，则在对应的通知栏中显示对应非法信息提示。
 - d) `login` 方法，参数为 `mouseevent`，实现与 GUI 中的 `login` 绑定，一旦点击则调用该方法。该方法实现以下功能：从用户名和密码输入框中获取输入信息，并将其做对应封装，即在字符串首部加入表 1.1 所示的相应编号，将其加入到发送服务器消息队列。
 - e) `signUp` 方法，参数为 `mouseevent`，实现与 GUI 中的 `signup` 绑定，一旦点击则调用该方法，将界面切换到注册界面。
 - f) `Modify` 方法，参数为 `mouseevent`，实现与 GUI 中的 `modify` 绑定，一旦点击则调用该方法，将界面切换到修改密码界面。

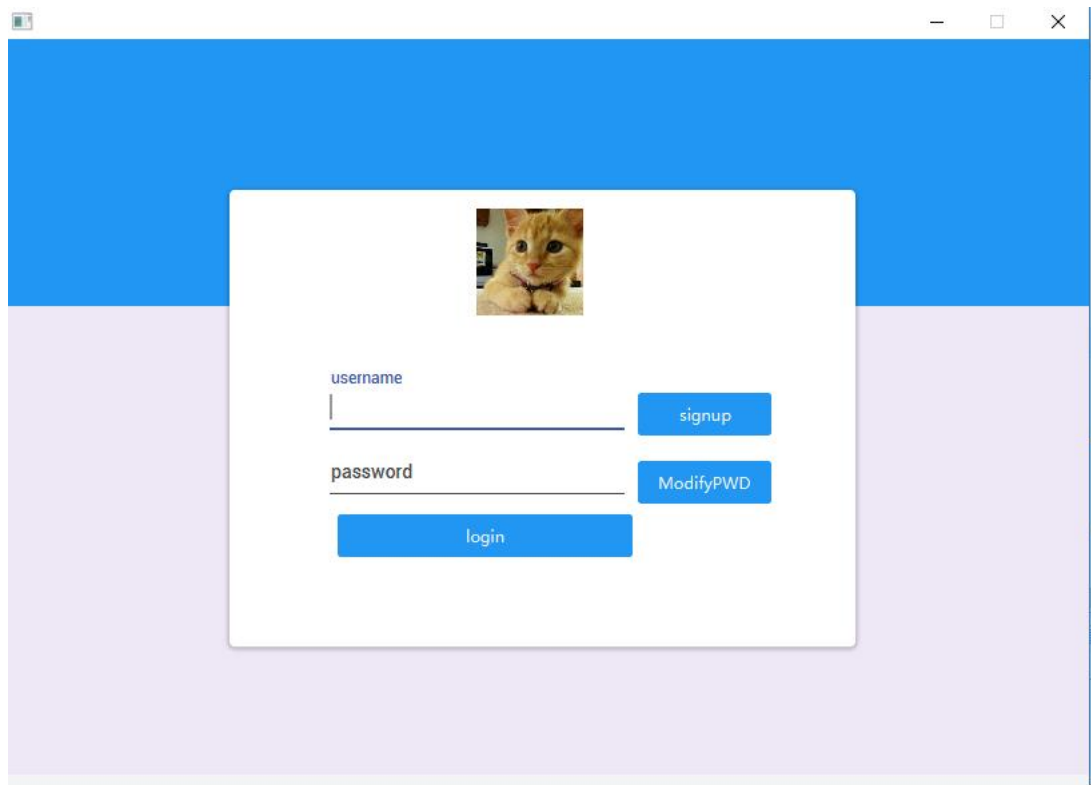


图 1.13 登陆界面效果图

- 3) 聊天界面实现，首先在 JavaFX 中画出需要的图形，生成 FXML 文件，效果图如图 1.14 所示。加入 `fx:controller="wxs.org.login.MainWindow"` 属性，并在对应按钮属性中加入 `onMousePressed` 属性，然后在对应路径中新建 `MainWindow.java` 文件，构造对应方法。
- a) `setInfo` 方法：初始化相关参数。
 - b) `add` 方法：聊天窗口的交换时储存添加当前聊天窗口的信息到对应的队列当中临时储存。
 - c) `getMessageFromScene` 方法：从聊天窗口的输入栏当中获取消息，并将其添加到发送消息队列。
 - d) `mark` 方法：判断当前储存的未显示到聊天窗口的信息是否包括对应用户信息，如果没有，则新建该用户对应的数据。
 - e) `sweep` 方法：情况储存的该用户的消息。
 - f) `initialize` 方法：初始化对应数据。
 - g) `setUserInfo` 方法：在聊天窗口的对应位置显示在线和离线用户消息及在线和离线用户数量。

-
- h) **show** 方法：在对应位置显示消息。
 - i) **createHbox** 方法：将传入的字符串转换为 **HBox** 类型，以便于显示。
 - j) **online** 方法，参数为 **mouseevent**，实现与 GUI 中的在线用户绑定，一旦点击则调用该方法。该方法实现以下功能：首先储存当前窗口显示内容，然后将其清空，判断是否有对应该在线用户消息，如果有则显示，如果没有则不处理。
 - k) **offline** 方法，参数为 **mouseevent**，实现与 GUI 中的离线用户绑定，一旦点击则调用该方法。该方法实现以下功能：首先储存当前窗口显示内容，然后将其清空，判断是否有对应该离线用户消息，如果有则显示，如果没有则不处理。
 - l) **refreshImgViewPressedAction** 方法：参数为 **mouseevent**，实现与 GUI 中的刷新按钮绑定，一旦点击则调用该方法。该方法实现以下功能：首先清空在线和离线用户显示列表，然后向服务器请求新的在线和离线用户。
 - m) **send** 方法，参数为 **mouseevent**，实现与 GUI 中的发送按钮绑定，一旦点击则调用该方法。该方法实现以下功能：首先获取对应消息，然后加上表 1.1 对应编号信息和相关用户名，将其加入到向服务器发送消息队列当中，并将发送栏清空，将其显示在上方窗口。
 - n) **transfer** 方法，参数为 **mouseevent**，实现与 GUI 中的发送按钮绑定，一旦点击则调用该方法。该方法实现以下功能：首先调用 **windows API** 弹出选择框，让用户选择要发送的文件，然后将对应文件信息和用户加入到发送给服务器队列。



图 1.14 聊天界面效果图

- 4) **changewindow** 线程主要是实现的是客户端几个界面的切换，如登录界面到聊天界面的切换，登录失败时提示登陆失败的 UI 实现等。主要是监视 **task** 队列，一旦发现有对应的任务，便将其取出，然后根据任务的类型调用不同的函数，如如果是登陆成功，则调用 **mainWindow** 中的 **setScene** 方法，切换到聊天界面；如果是登陆失败，则调用 **login** 中的 **notification** 方法，提示用户登陆失败原因等。该消息的主要来源是 1.4.2 中所述的与服务器通信线程。
- 5) 服务器端登陆界面套用客户端登陆界面模板，设计思路一致，直接使用了之前设计的 UI，在对应的控制代码中添加相关方法即可，这里不再赘述。
- 6) 服务器显示 UI 实现，首先在 JavaFX 中画出需要的图形，生成 FXML 文件，效果图如图 1.15 所示。加入 **fx:controller="wxs.org.Server.serverController"** 属性，并在对应按键属性中加入 **onMousePressed** 属性，然后在对应路径中新建 **ServerController.java** 文件，构造对应方法。
 - a) **show** 方法：获取在线和离线用户消息，并按照顺序将其显示在图形界面对应的位置。
 - b) **update** 方法，参数为 **mouseevent**，实现与 GUI 中的刷新按钮绑定，一旦点击则调用该方法。该方法实现以下功能：清空界面显示内

容，然后在任务队列当中添加刷新请求，告知对应线程需要获取最新用户信息。

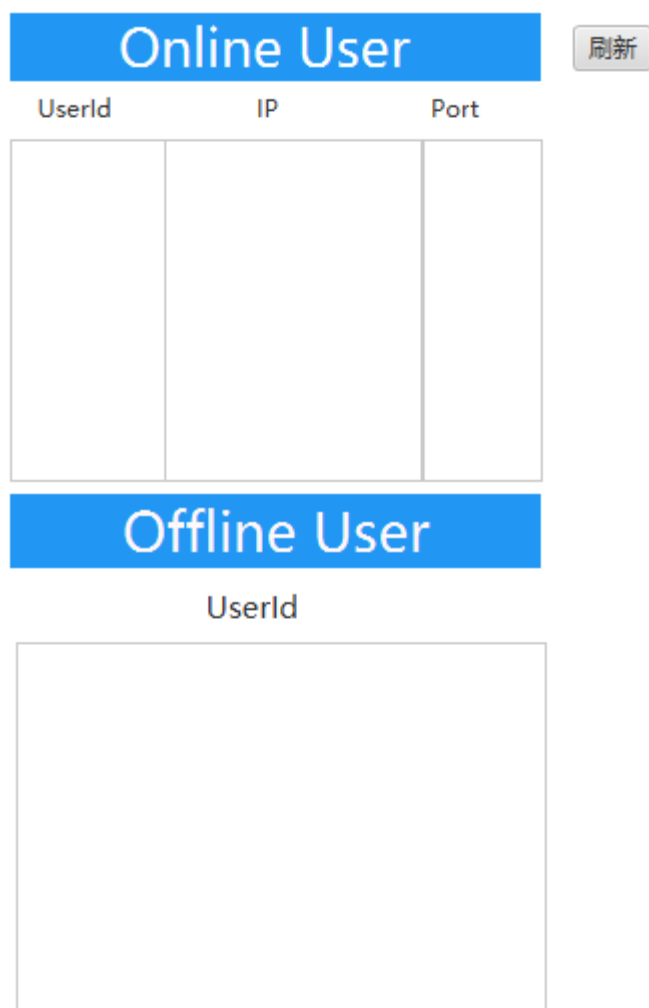


图 1.15 服务器显示界面

1.4.5 数据库模块实现

- 1) 数据库采用 mysql，使用插件为 mysql-connector-java-5.1.38.jar，主要对 sql 语句进行封装，实现对数据库的增、删、查、改四大功能。
- 2) Creatdb 类主要封装了创建数据库的功能，首先通过 getConnection 方法与数据库建立连接，然后使用 sql 语句进行判断是否已经存在对应的表，如果不存在则创建对应的空表。在数据库当中主要创建四个表，分别储存注册用户信息、在线用户信息、离线用户信息和离线消息。以创建注册用户信息为例，使用如下 sql 语句进行创建，
"CREATE TABLE IF NOT EXISTS USRINFO " + "(usrId

VARCHAR(255) not NULL, " + " psw VARCHAR(255), " + " question
VARCHAR(255), " + " answer r VARCHAR(255), " + "ip
VARCHAR(255), " + "port VARCHAR(255), " + " PRIMARY KEY
(usrid));", 创建成功时, 可以使用 navicat 查看数据库中表项, 发现
新增了一个 chat 数据库, 含有 message、offline、online 和 usrinfo 四
个表, 如图 1.16 所示。

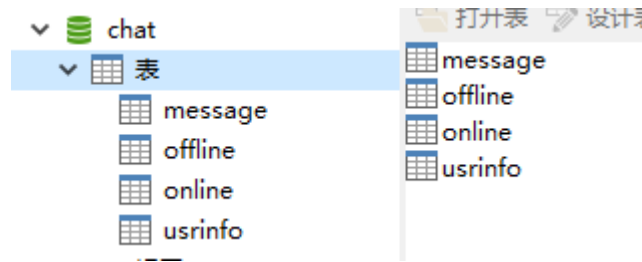


图 1.16 数据库成功创建

- 3) Database 类主要实现与数据库的交互, 主要实现了一下方法与数据库进行交互:
- a) get()方法, 无参数, 返回类型为 ArrayList<ArrayList<String>>, 返回注册用户信息, 包括用户名、ip 地址和端口号。主要使用 sql 语句中的 select 语句, 从数据库中获取后进行解析。
 - b) insert 方法, 参数为字符串数组, 返回值为 boolean 类型, 根据 boolean 值来判断向哪个表中插入, 为真时, 向数据库中用户注册信息表插入, 插入信息为字符串数组内容。返回值为真时表示插入成功, 返回值为假时表示插入失败。主要使用 sql 语句中的 insert 语句, 向数据库中插入信息。
 - c) update 方法, 参数为字符串数组和 boolean 值, 无返回值。主要更新用户信息中的密码、ip 地址和端口号。当 boolean 值为真时, 通过字符串数组中的信息, 更新特定用户密码信息; 当 boolean 值为假时, 通过字符串数组中的信息, 更新特定用户的 IP 地址和端口号。主要使用 sql 语句中的 update 语句, 更新数据库中对应信息。
 - d) update 方法, 重载 update, 只有一个字符串数组, 进行更新离线消息, 首先判断离线消息当中是否存在该对应用户发送的离线消息, 如果存在则在后面添加, 如果不存在则在表中插入该离线消息。主要使用 sql 语句中的 update 和 insert 语句, 更新数据库中对应信息。
 - e) search 方法, 参数为一个字符串, 返回值为一个 list。主要通过字

符串信息，对 `usrinfo` 表进行查询，然后将用户名、密码、密保问题、密保答案、ip 地址和端口号加入到 `list` 当中，将其返回。主要使用 `sql` 语句中的 `select` 语句，对数据库进行查询。

f) `search` 方法，参数为一个字符串数组，重载 `search` 方法，返回一个 `list`。主要通过字符串数组中的接收者和发送者信息进行查询，然后将其离线信息加入到 `list` 中并返回。主要使用 `sql` 语句中的 `select` 语句，对数据库进行查询。

g) `getMessage` 方法，参数为一个字符串，返回值为一个 `list` 的 `list`。主要通过字符串信息，对 `message` 表进行查询，然后该 `string` 对应的每个离线消息加入到 `list` 当中，将每一个 `list` 加入到 `list` 的 `list` 当中，将其返回。主要使用 `sql` 语句中的 `select` 语句，对数据库进行查询。

h) `delete` 方法，参数为一个字符串和一个 `boolean` 值，无返回值。`boolean` 值为真时，通过用户名删除注册用户信息表中对应的信息，`boolean` 值为假时，通过用户名删除离线消息表中对应的信息。主要使用 `sql` 语句中的 `delete` 语句，对数据库对应的表项进行删除操作。

i) `exchange` 方法，参数为一个字符串和一个 `boolean` 值，无返回值。主要实现对应用户上下线信息的交换，具体实现为，当 `boolean` 值为真时在 `offline` 表中删除该用户，并将其添加到 `online` 表中；当 `boolean` 值为假时在 `online` 表中删除该用户，并将其添加到 `offline` 表中。主要使用 `sql` 语句中的 `delete` 和 `insert` 语句，对数据库对应表项进行删除和插入操作。

j) `insert_offline` 方法，参数为一个字符串，无返回值。主要是将该用户名添加到 `offline` 表中，主要在注册成功时使用该函数，使用 `sql` 语句中的 `insert` 语句，对数据库对应表项进行插入操作。

k) `get` 方法，参数为一个字符串，重载之前的 `get` 方法，返回值为 `list`，输入应为 `offline` 或者 `online`，获取对应表中的所有信息。主要使用 `sql` 语句中的 `select` 语句，对数据库对应表项进行查找操作。

l) `deleteMessage` 方法，参数为一个字符串，无返回值。删除 `message` 表项中储存的对应用户的离线消息。主要使用 `sql` 语句中的 `delete` 语句，对数据库对应表项进行删除操作。

- 4) 以用户信息为例，将注册用户信息加密后储存到数据库中 `message` 表中，储存格式如图 1.17 所示。

usrid	psw	question	answer	ip	port
al2jRSM7FnU=	QMlg6J1bsKI=	al2jRSM7FnU=	al2jRSM7FnU=	/AoMry5n3RwVoVHQexEnFw==	g3SvAQdCdcA=
H+PHKXV+SDA=	H+PHKXV+SDA=	H+PHKXV+SDA=	H+PHKXV+SDA=	/AoMry5n3RwVoVHQexEnFw==	aJfENr4TtA=
I2RsUMr2kd8=	I2RsUMr2kd8=	I2RsUMr2kd8=	I2RsUMr2kd8=	/AoMry5n3RwVoVHQexEnFw==	aJfENr4TtA=
QMlg6J1bsKI=	QMlg6J1bsKI=	QMlg6J1bsKI=	QMlg6J1bsKI=	dXrS+/qH6lJfwftjUinR1A==	aJfENr4TtA=
▶xthTJVnT/ak=	xthTJVnT/ak=	xthTJVnT/ak=	xthTJVnT/ak=	HGWWmgguLPv5b5yyruJVqg==	sU0wRDnEZPs=

图 1.17 数据库中注册用户信息表

1.4.6 加密模块实现

- 1) 加密算法采用 DEC 加密算法，相关依赖库为
org.apache.commons.codec.binary.Base64，读入参数为字符串，返回加密后的字符串。
- 2) 解密算法为加密算法的逆操作，相关依赖库为
org.apache.commons.codec.binary.Base64，读入参数为字符串，返回解密后的字符串。

1.5 系统测试及结果说明

1.5.1 测试环境

1. 硬件环境

windows10 家庭中文版；处理器：Inter(R) Core(M) i5-4210U CPU @ 1.70GHz 2.40GHz；内存：8.00 GB(7.89 GB 可用)；系统类型：64 位操作系统。

VMware® Workstation 12 Pro 12.5.7 build-5813279；内存：2G；处理器：Inter(R) Core(M) i5-4210U CPU @ 1.70GHz 2.40GHz（2 处理器）；硬盘：60GB；网络适配器：NAT；Windows 10 专业版。

2. 网络配置

名称： WLAN

描述： Qualcomm Atheros AR956x Wireless Network Adapter

物理地址(MAC): ac:e0:10:33:19:8f

状态： 可操作

最大传输单元: 1500

链接速度(接收/传输): 144/72 (Mbps)

DHCP 已启用: 是

DHCP 服务器: 10.14.127.254

DHCP 租约获得时间: 2017年12月22日 23:42:34

DHCP 租约到期时间: 2017年12月23日 1:42:34

IPv4 地址: 10.14.126.69/22

IPv6 地址: 2001:250:4000:8041:586e:ce8b:88d3:90a2/64,
2001:250:4000:8041:a1e5:492f:25ab:2f19/128, fe80::586e:ce8b:88d3:90a2%20/64

默认网关: fe80::1614:4bff:fe7d:4cbd%20, 10.14.127.254

DNS 服务器: 202.114.0.242, 202.114.0.131

DNS 域名:

DNS 连接后缀:

DNS 搜索后缀列表:

网络名称: HUST_WIRELESS 2

网络类别: 公共

连接性(IPv4/IPv6): 已连接到 Internet

3. 运行环境

软件组件: Eclipse Oxygen Release (4.7.0), JDK 1.8.0_144,
MySQL5.6.36-log。

第三方组件: mysql-connector-java-5.1.38.jar,
sun.misc.BASE64Decoder.jar, sun-jndi-ldap.jar, jfoenix.jar。

1.5.2 测试结果

1. 进入到 `wxs.org.Server` 中的 `Main` 类, 运行该类的 `main` 方法, 启动服务器端程序, 控制台输出相关状态信息, 如图 1.18 所示, 并且弹出服务器端登陆界面, 如图 1.19 所示。此外, 由于考虑到服务器端很少会关闭, 这里将界面程序与后台运行程序隔离开来, 以避免操作者不小

心点击关闭界面按钮关闭服务器程序，导致用户连接失败，不能正常进行聊天，带来不好的用户体验。所以，在测试之前需要进入到 `wxs.org.Server` 中的 `ServerStart` 类，运行该类中的 `main` 方法，启动服务器后台 运行程序。

```
Connecting to database...  
Creating database...  
Database created successfully...
```

图 1.18 启动服务器时控制台输出信息

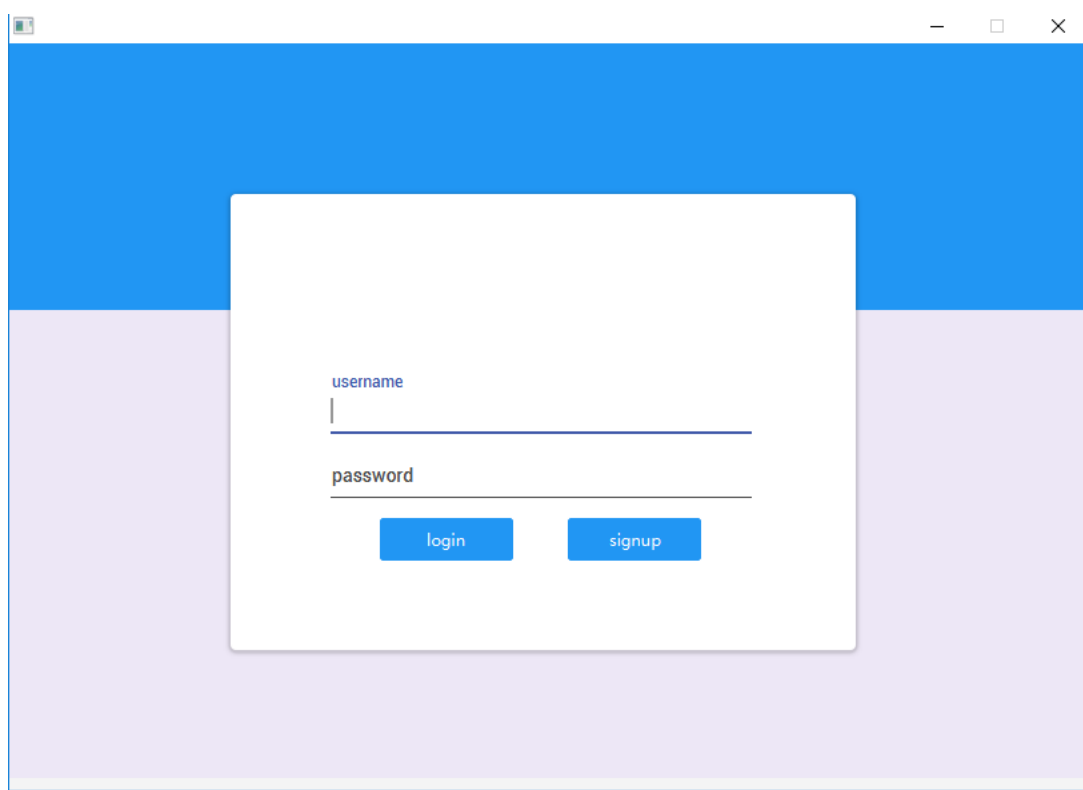


图 1.19 服务器端登陆界面

2. 输入账号密码，然后点击 `login`，如果输入账号名过短，则会在输入用户名上面的提示框里输出提示信息“`username is too short`”，提示用户用户名过短，如图 1.20 所示；如果密码过短则会在与图 1.20 同样的提示框中提示用户“`password is too short`”，如图 1.21 所示；如果账号或者密码输入错误，则登陆失败，并且在会与图 1.20 同样的提示框中告诉用户登陆“`Incorrect password or userid`”，如图 1.22 所示。输入正确的账号密码，则可以登陆成功，跳转到显示用户信息界面，如图 1.23 所示。可以看到，界面主要分为上下两部分，上方显示在先用户，下方显示离线用户，由于此时刚打开服务器，没有用户上线，所

以在线用户显示为空，而离线用户显示有如图 1.23 中的六个用户。右上方还有一个刷新按钮，用来更新当前的在线和用户状态，此测试将在 4 及以后相关部分进行测试。（注：默认帐户名为：admin，密码为 admin2017）

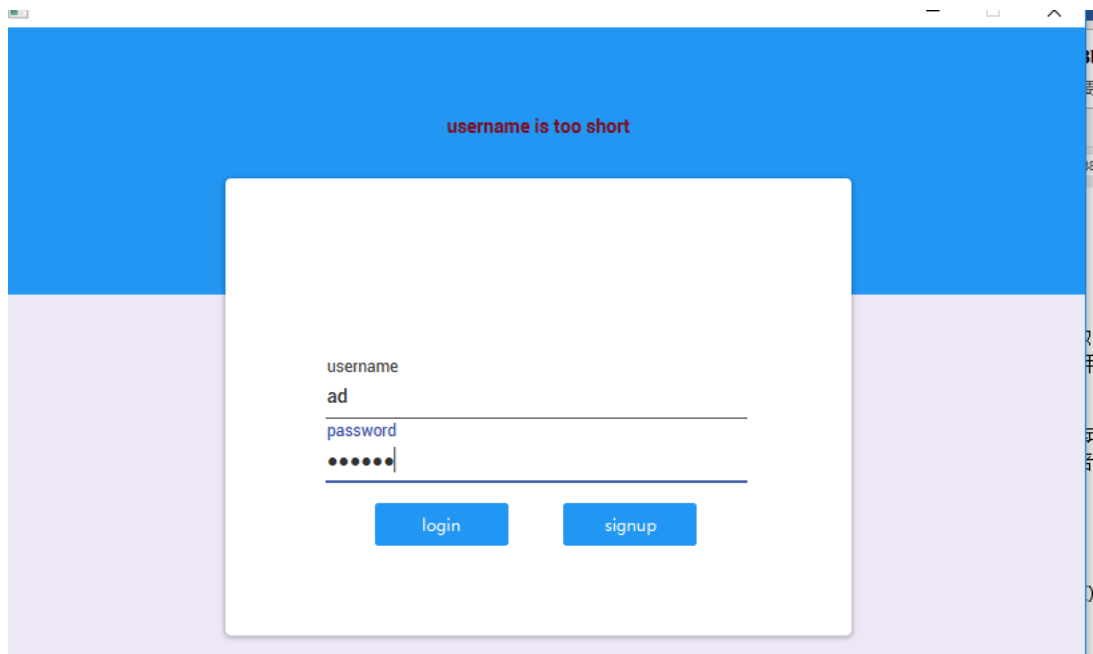


图 1.20 用户名过短提示消息

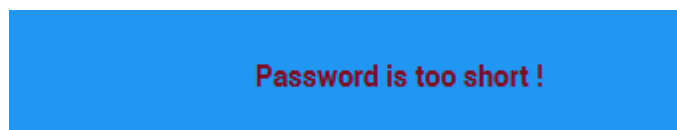


图 1.21 密码过短提示消息

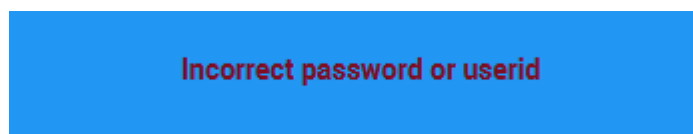


图 1.22 密码错误提示消息

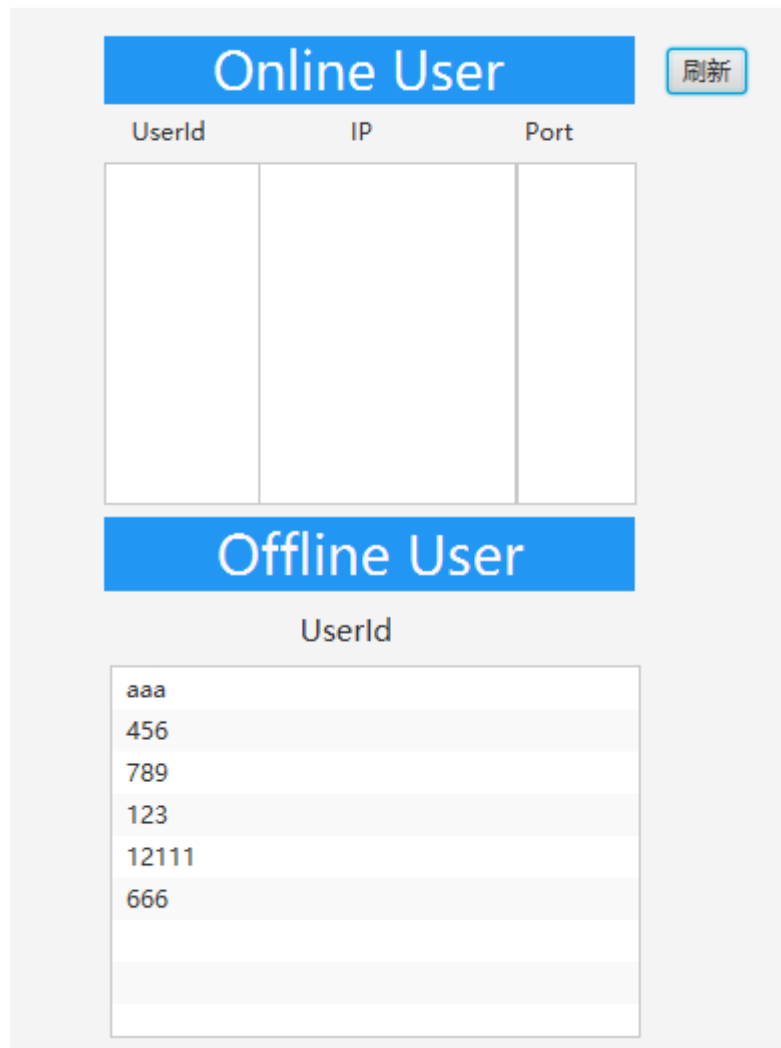


图 1.23 显示用户信息界面

3. 进入到 wxs.org.login 的 Main 类，运行该类的 main 方法，启动服务器端程序，控制台输出相关状态信息，如图 1.24 所示，并且弹出服务器端登陆界面，如图 1.25 所示。

```
port10001
set user info
mainwindow online[]
mainwindow offline[]
offline + []
ChatController wxs.org
```

图 1.24 启动客户端控制台输出信息

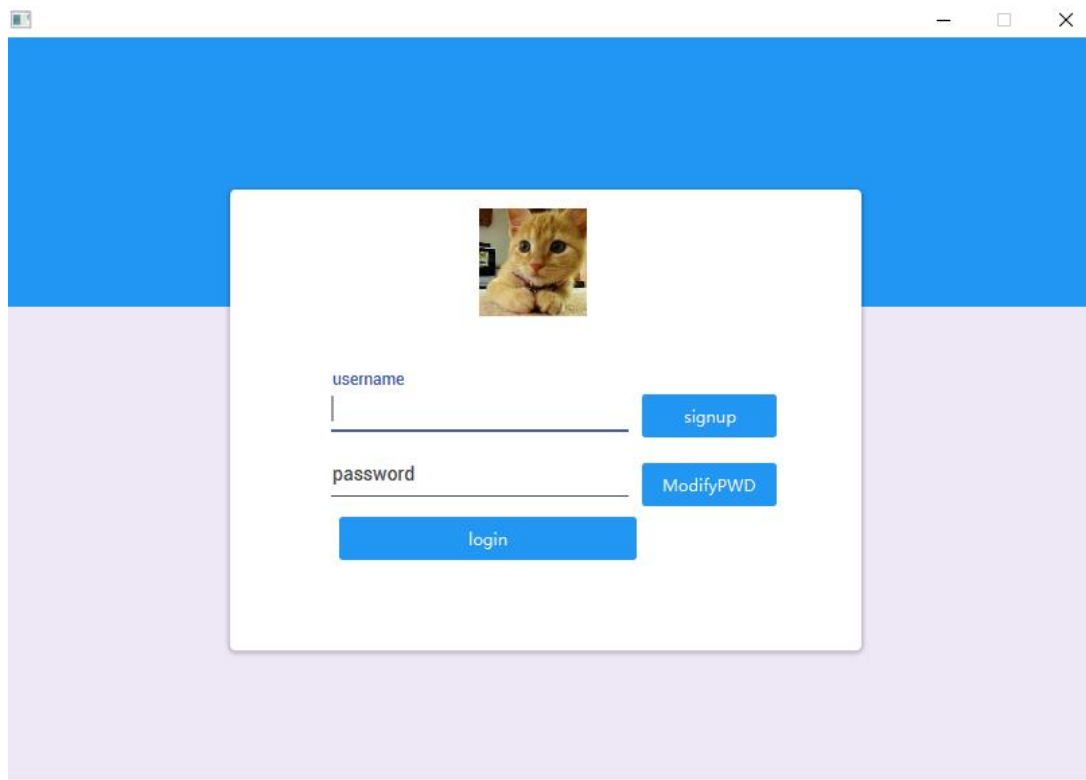


图 1.25 启动客户端后登陆界面

4. 如果输入用户信息非法或者输入帐户名密码错误，依旧会如 2 中所述提示对应信息，这里不再一一赘述。点击 **signup** 按钮，将会将会跳转到注册界面，如图 1.26 所示。点击 **back** 按钮，可以返回到登陆界面。现在测试注册功能，2 中我们在服务器端界面已经看到了用户名为 123 的用户，这里我们先注册一个用户名为 123 的用户，测试是否能够注册相同用户名的用户。测试结果如图 1.26 所示，可以看到当输入用户名为 123 的用户注册信息时，会提示用户输入用户名错误。然后我们注册一个不存在的用户名，这里测试为 1234，测试结果如图 1.27 所示。可以看到，输入 1234 用户名的相关信息后，注册成功，跳转至登陆界面，此时切换到服务器端显示用户信息界面，点击刷新按钮，结果如图 1.27 所示。与图 1.23 相比，离线用户信息，多了一个用户名为 1234 的用户。此时切换到客户端显示用户信息界面，点击刷新按钮，可以看到在线用户栏中显示两个在线用户，分别为 123 和 1234，并显示了其对应的 IP 地址和端口，并且离线用户对应部分不再显示对应用户，如图 1.28 所示。

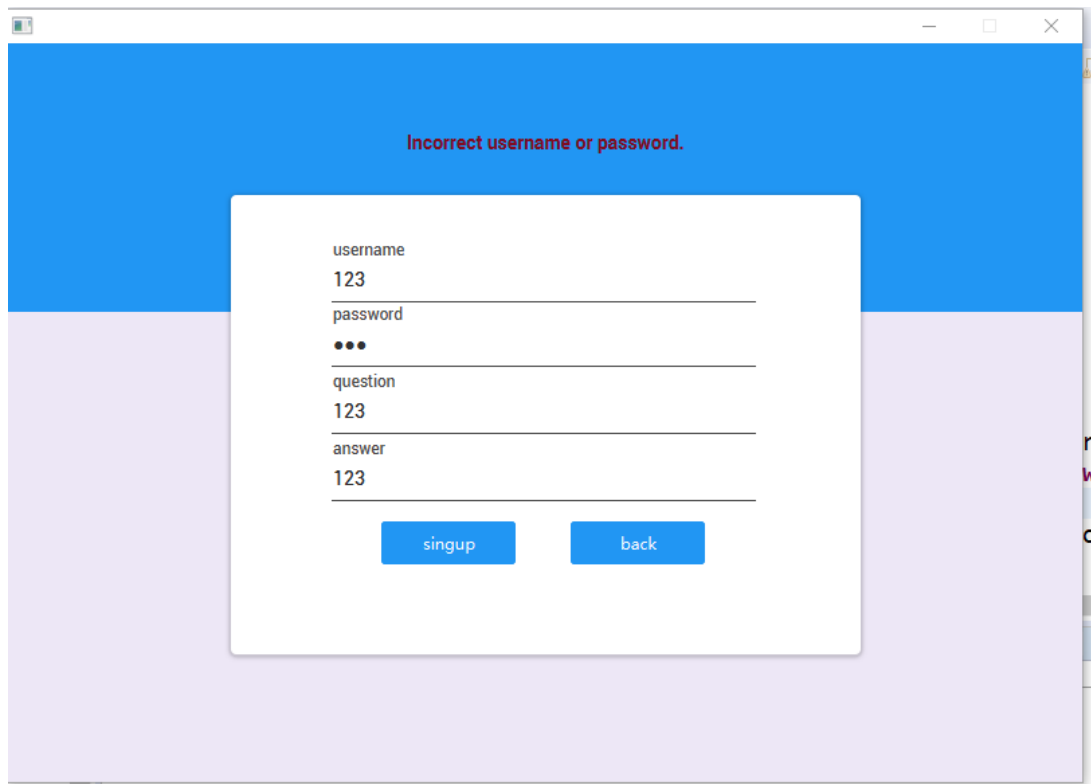


图 1.26 输入用户名已存在的注册信息

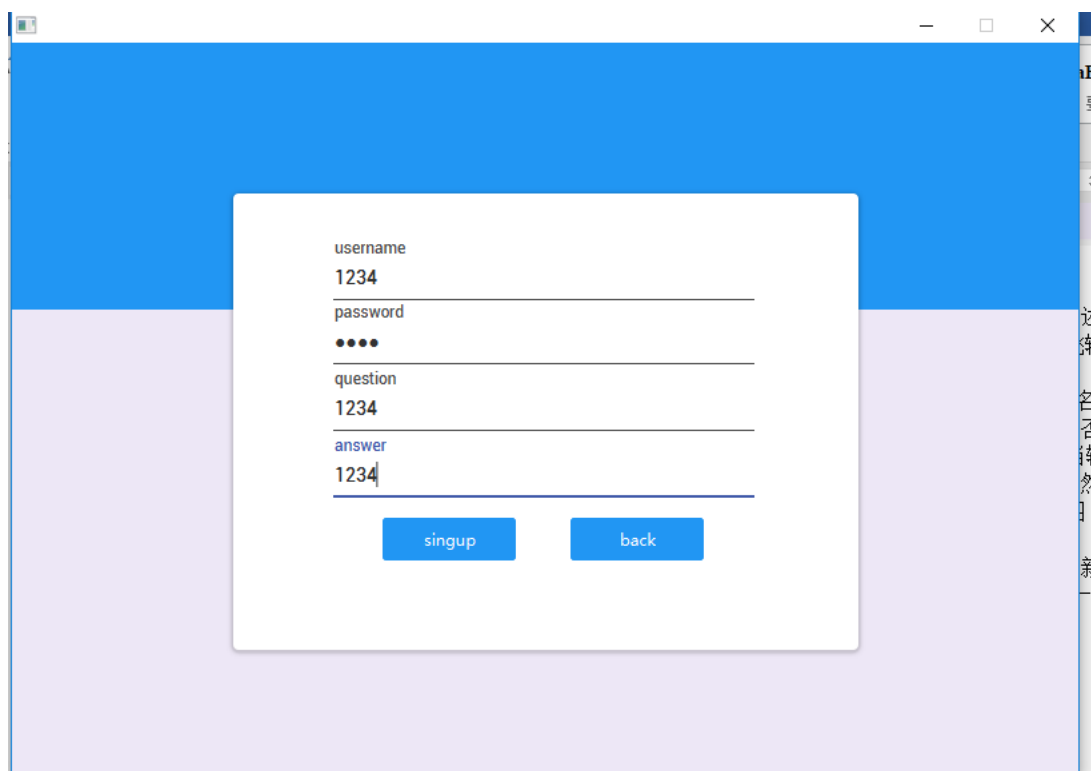


图 1.27 注册新用户测试

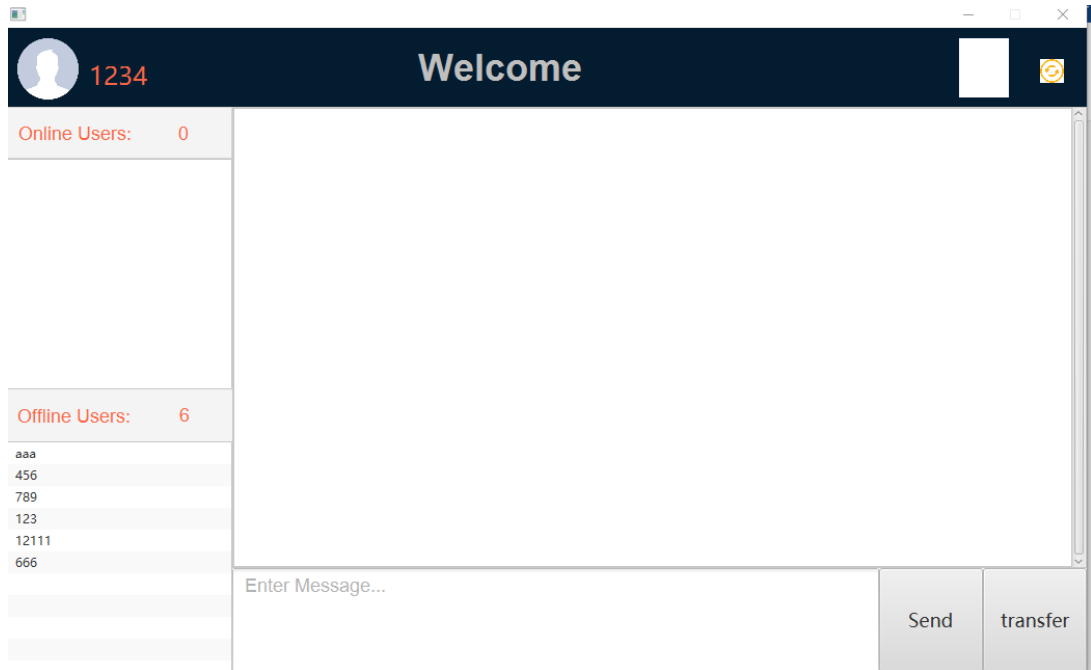
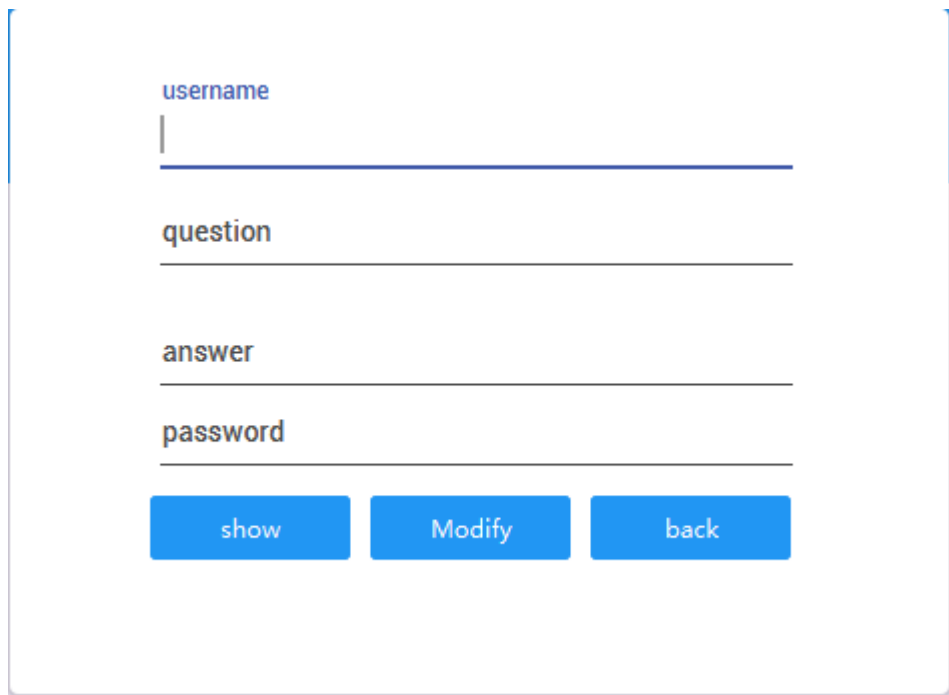


图 1.29 登陆成功后聊天界面

- 退出当前用户 1234，开启新的客户端，点击登陆界面的 **modify** 按钮，进入修改密码界面，如图 1.30 所示。首先输入需要修改的用户名，然后点击 **show** 按钮，显示该用户注册时设置的密保问题，原先设置的密保答案为 1234，这里输入 123，可以看到会提示问题错误，如图 1.31 所示。然后输入正确答案 1234，将密码修改为 123，修改成功后自动跳转到登陆界面，如图 1.32 所示。此时便可以使用新的密码完成登陆。



A screenshot of a web interface for modifying a password. It features four input fields labeled 'username', 'question', 'answer', and 'password'. The 'username' field contains a vertical cursor. Below the fields are three blue buttons: 'show', 'Modify', and 'back'.

username
|

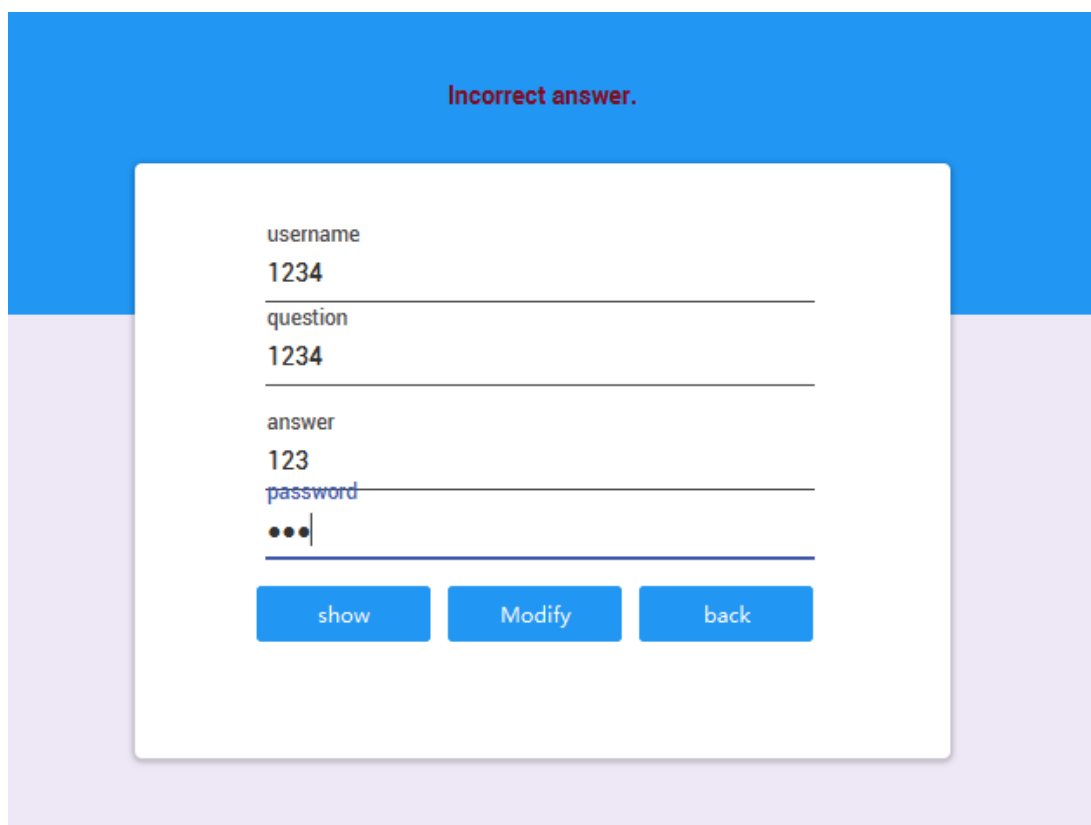
question

answer

password

show Modify back

图 1.30 修改密码界面



A screenshot of a web interface showing a password modification failure. A blue banner at the top displays the text 'Incorrect answer.' in red. Below the banner is a white modal box containing the same form as in Figure 1.30. The 'username' field contains '1234', the 'question' field contains '1234', the 'answer' field contains '123', and the 'password' field contains three dots. The 'show', 'Modify', and 'back' buttons are at the bottom.

Incorrect answer.

username
1234

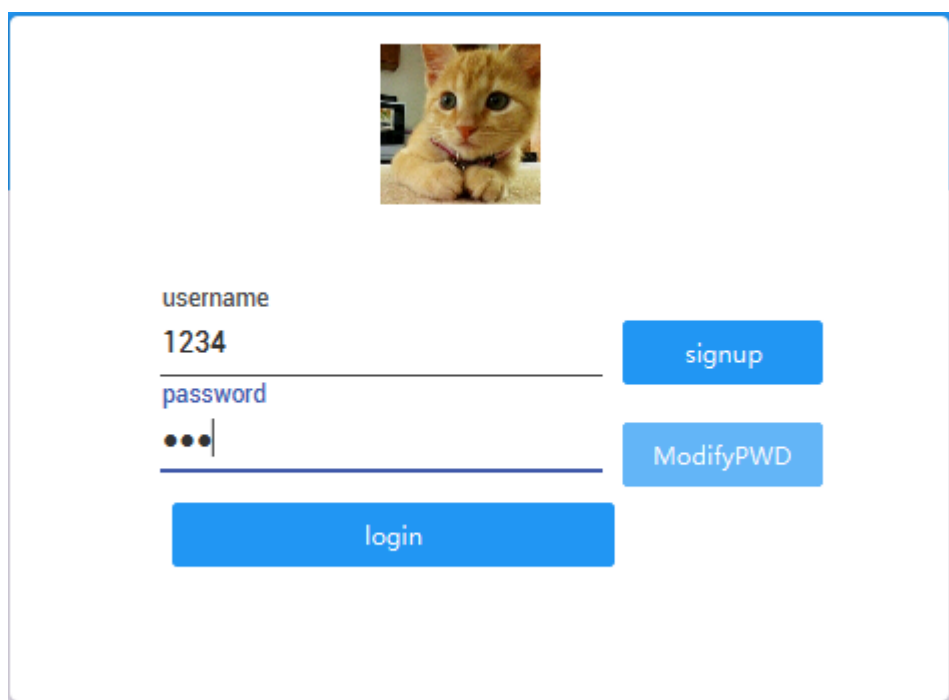
question
1234

answer
123

password
●●●

show Modify back

图 1.31 修改密码失败界面



A login form interface. At the top center is a profile picture of an orange cat. Below it, the text 'username' is followed by a text input field containing '1234'. To the right of this field is a blue button labeled 'signup'. Below the username field is the text 'password' followed by a password input field with four black dots and a cursor. To the right of this field is a blue button labeled 'ModifyPWD'. At the bottom center is a large blue button labeled 'login'.

图 1.32 使用新密码进行登陆

7. 再次启动一个客户端，输入用户名 123，完成登陆，然后与用户 1234 发送消息，如图 1.33 所示。用户 1234 点击刷新按钮，可以看到此时在线用户有用户 123，然后点击用户 123，打开与用户 123 的聊天窗口，可以看到用户 123 发送的消息，如图 1.34 所示。



图 1.33 用户 123 向用户 1234 发送消息

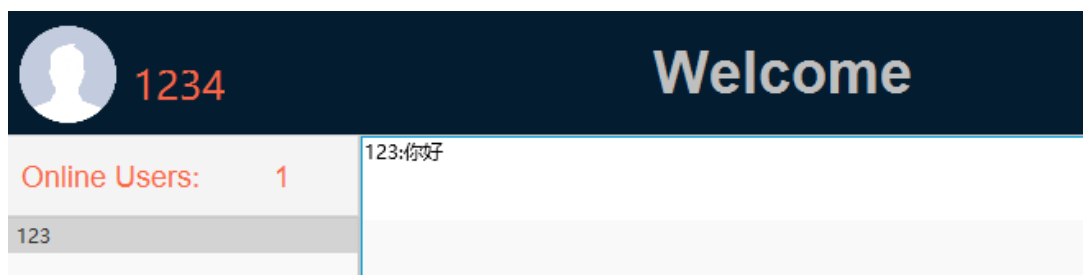


图 1.34 用户 1234 接收到用户 123 发送的消息

8. 两个用户之间互发消息，可以看到界面可以及时的收到并且显示对应消息，达到点对点发送消息功能，符合设计要求，结果如图 1.35 所示。



图 1.35 发送消息显示图

9. 使用用户 123 向离线用户 456 发送离线消息，如图 1.36 所示。此时离线消息应该储存到服务器端的数据库当中。打开 Navicat Premium，连接本地数据库，打开 chat 数据库中的 message 表可以看到已经储存了离线消息，但是由于采取了加密，所以数据库中没有明存储该信息，如图 1.37 所示。

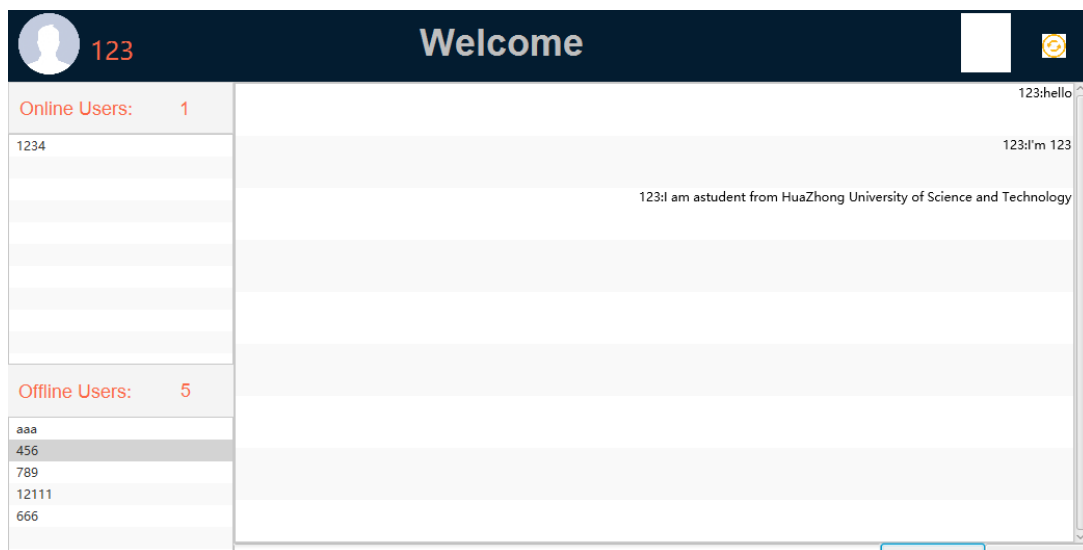


图 1.36 用户 123 向用户 456 发送离线消息

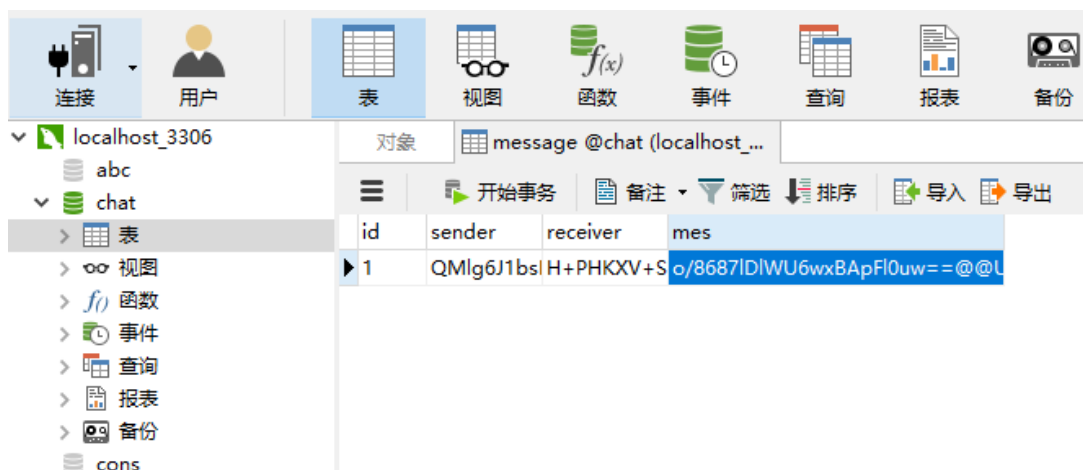


图 1.37 数据库中储存加密后的离线消息

10. 在虚拟机上登陆用户 456，登陆成功后点击用户 123，进入与用户 123 的聊天界面，可以看到用户 123 发送的离线消息，如图 1.38 所示，此时可以继续和 123 发送聊天消息，进行聊天，如图 1.39 所示。说明离线消息工作正常，并且可以保证多个用户同时在线并且互发消息，符合设计要求。

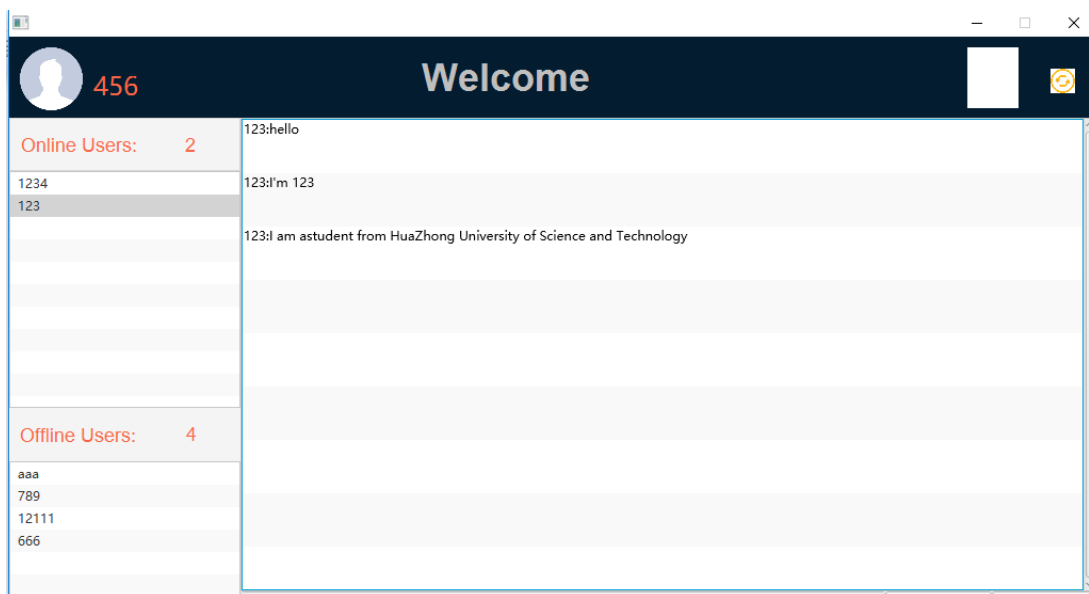


图 1.38 用户 456 接收到用户 123 发送的离线消息

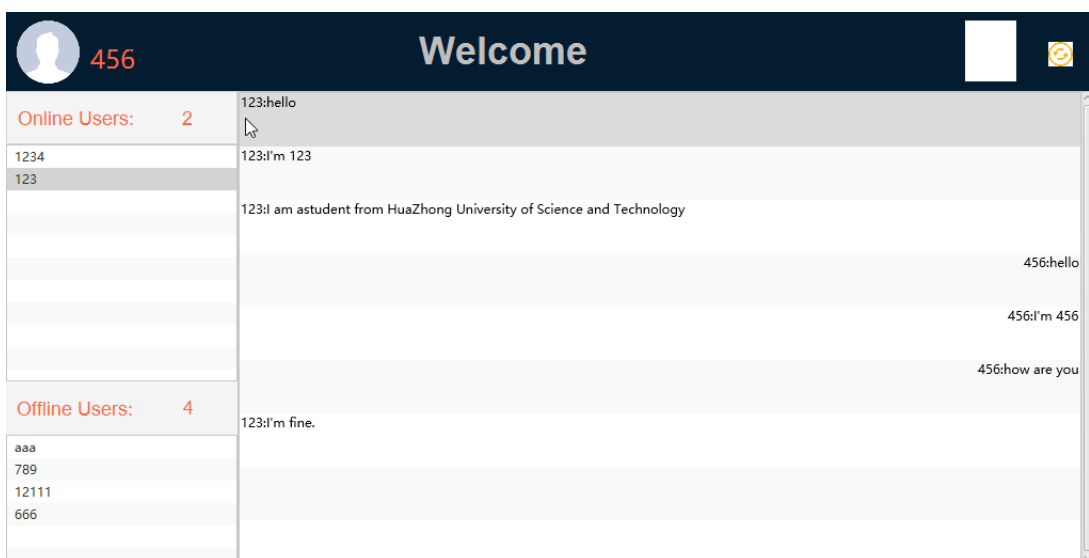


图 1.39 用户 456 与用户 123 在线通信

11. 使用用户 456 向用户 123 发送测试文件，首先点击 **transfer** 弹出选择文件路径，选取测试目录下的[了不起的狐狸爸爸].Fantastic.Mr.Fox.2009.BD720P.X264.AAC.HALFCD-NORM.mkv 文件，如图 1.40 所示。然后点击确定，即可开始发送，可以在界面的右上角空白处实时看到传输速率，并且传输速率会以输出到 D:/junior/network 中，生成 excel 表格，以便作图分析发送完成后即可在 D:/junior/network 文件夹中找到对应文件，打开可以正常播放，并且大小一直，说明发送文件是可靠的。

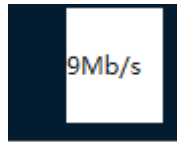


图 1.40 发送过程中速率显示



图 1.41 接收完成后可以正常播放

12. 根据生成的 excel 表格做出速率曲线图如图 1.42 所示，可以看到传输过程中波动较大，且峰值达到了 13MB/s，这可能是因为选取的时间间隔为 0.1s，相对来讲比较小，可能在某些 0.1s 甚至更长的时间内，主要在发送包，而在另一段时间内主要在等待 ACK，所以导致单次的传输速率计算不准确，导致图形波动较大，如果对每一段取平均值，可以看出，刚开始传输速率缓慢上升，这是由于刚开始窗口没有被全部占用，导致不能达到较好的效果，随着时间的推进，窗口的利用率逐渐增大，所以平均来看其传输速率较高，而在最后有下降趋势，这是由于最后只是等待 ACK，没有更多新的包的发送，所以有所下降。网卡速率未 72.2Mbps，根据图表做出合理估计，平均传输速率约为 8.2MB/s，网络带宽利用率约为 91%，可以看到，传输文件的效率还是比较高的，说明该实现相对来讲还是比较完善。

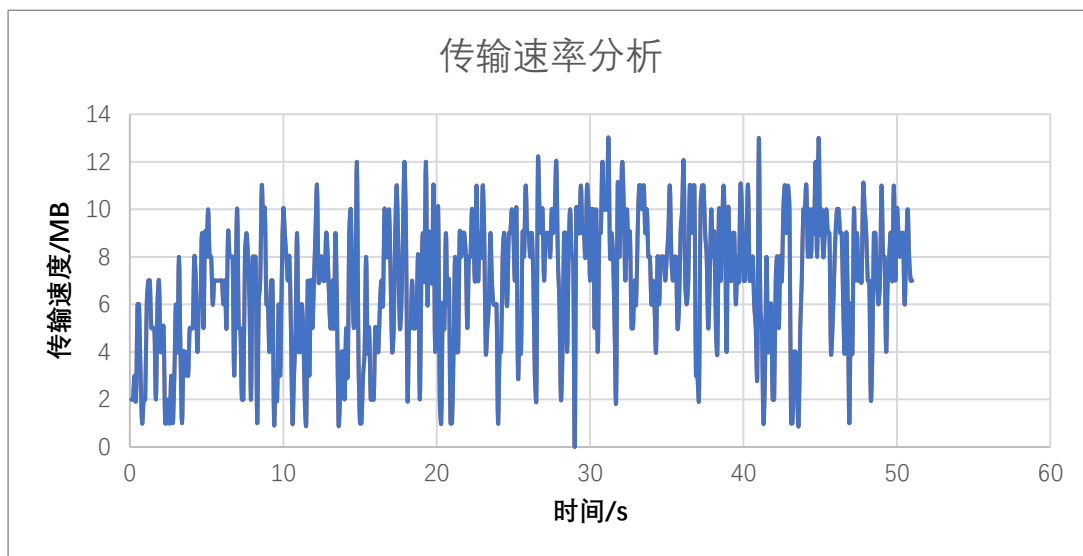


图 1.42 传输文件速率图

13. 进入 Navicat Premium，连接数据库，打开 chat 数据库中 userinfo 表，查看数据库中存储的用户信息，可以看到所有的信息都是加密之后存储的，没有出现明文存储的情况，如图 1.43 所示，其他三个表类似，说明实现了加密技术。

userid	psw	question	answer	ip	port
al2jRSM7FnU=	QMlg6J1bsKI=	al2jRSM7FnU=	al2jRSM7FnU=	/AoMry5n3RwVoVHQexEnFw==	g3SvAQdCdcA=
H+PHKXV+SDA=	H+PHKXV+SDA=	H+PHKXV+SDA=	H+PHKXV+SDA=	C2rdiXCpKuB8o82Y2RRCCA==	c5ipxN2XBDA=
I2RsUMr2kd8=	I2RsUMr2kd8=	I2RsUMr2kd8=	I2RsUMr2kd8=	/AoMry5n3RwVoVHQexEnFw==	aJfENr4TtA=
NpCWS+M+7Fg=	QMlg6J1bsKI=	NpCWS+M+7Fg=	NpCWS+M+7Fg=	oToqL4iXeqNr5rBGL//0aA==	vRQ6kW5rLq0=
QMlg6J1bsKI=	QMlg6J1bsKI=	QMlg6J1bsKI=	QMlg6J1bsKI=	C2rdiXCpKuB8o82Y2RRCCA==	aJfENr4TtA=
xthTJVnT/ak=	xthTJVnT/ak=	xthTJVnT/ak=	xthTJVnT/ak=	HGWWmgguLPv5b5yyruJVqg==	sU0wRDnEZPs=

图 1.43 数据库中 userinfo 储存信息

1.6 其它需要说明的问题

1. 对于文件发送，之前采用的是动态窗口，不断的增加窗口的大小，一旦发生丢包将窗口值设置为当前窗口值得一半加 3，门限值设置为当前拥塞窗口得一半；超时拥塞窗口设置为 1，门限值设置为当前拥塞窗口得一半。但是发现性能并不是很好，而且在发送大文件的时候由于拥塞窗口的不断增加会发生丢包，导致不必要得浪费。所以最后还是采取了静态窗口的方法，根据多次调参，选取了一个合适的窗口值，达到尽可能好的网络效率。由于网络设备的发展以及协议的成熟，当前网络中尤其是在我们的局域网测试环境当中，丢包的概率非

常小，所以设置为一个静态窗口可以避免不必要的丢包，更好的利用网络资源。

2. 注意本工程所使用的的所有相关插件都在对应文件夹内包括，如需运行请首先将所需插件加入，否则将会报错，无法正常运行。如本工程在运行过程中出现 bug，请联系本人做出相应修改。另外，根据测试，本工程可以兼容 windows7、windows8、windows10、Ubuntu 等系统，但是由于 Ubuntu 的文件路径与 windows 的文件路径分隔符不同，所以如果需要在不同系统下进行测试，建议将路径改为相对路径，以避免不必要的麻烦。测试时请尽量保证自己的相关软件版本高于或等于本报告中列出的版本，否则可能会出现问题。如果由于与本工程开发和运行版本不一致出现问题，请更新自己的软件至最新版本。
3. 如果使用的 IDE 没有安装 JavaFX，可能会无法运行该工程，因为该工程使用 JavaFX 作为 UI 开发，这里以 Eclipse 为例讲解如何安装 JavaFX。

a) 点击 Help 中的 Install New Software，如图 1.44 所示。

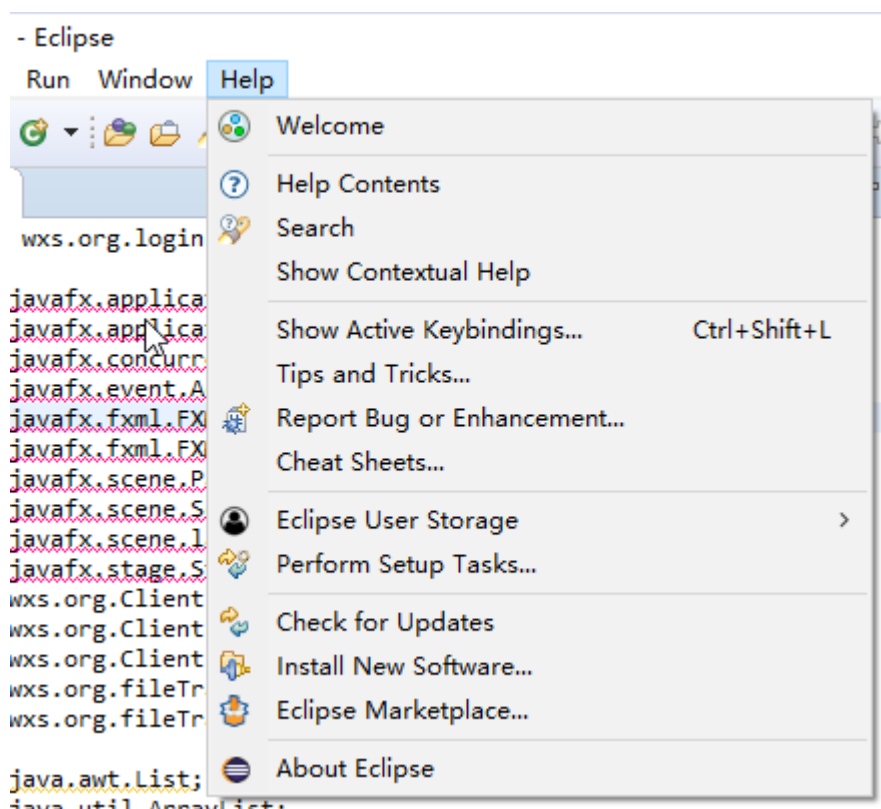


图 1.44 打开 Install New Software 示例

b) 点击后可以看到如图所示的界面，输入图中所示信息，然后选中

对应的插件，然后一直点击 Next，完成安装过程。

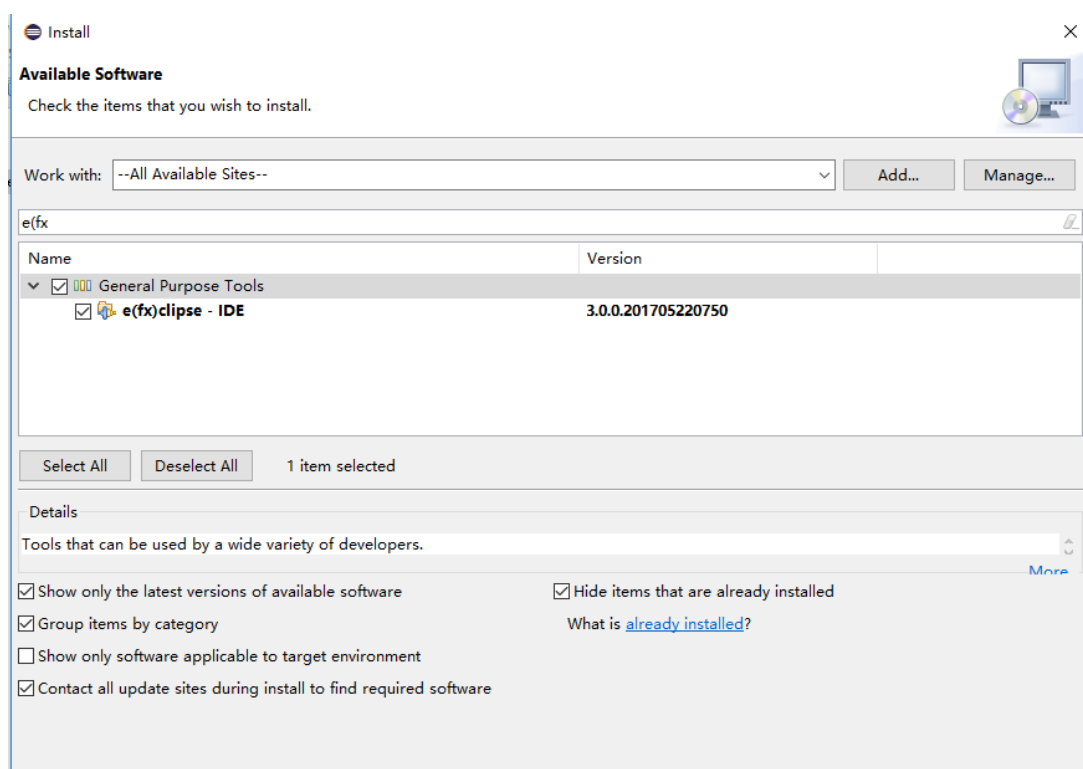


图 1.45 安装对应插件

4. 本工程服务器端需要连接 mysql 数据库，运行之前请确保自己的 PC 上安装有对应的数据库，默认帐户为 root，密码为 123456，如果该信息不符，会导致连接数据库失败，之后的测试将无法完成，如需测试请在 wxs.org.Db 包中的 Database 类中修改为本地数据库的账户密码，完成后续测试。
5. 由于不同机器上的默认编码格式不同，如果发送消息中文显示乱码，请调整自己的编码格式为 UTF-8，建议测试时使用英文，以避免出现这种问题，引起不必要的麻烦。
6. 本工程由本人独立开发，图形界面借鉴 <https://github.com/DomHeal/JavaFX-Chat>，在此特别感谢该开源作者。其他均为自己完成，如在使用过程中出现任何问题，请与本人联系，协助解决。本人拥有此工程的最终解释权，且其未开源，仅限于提交审核，如需用作他用途，请联系本人获得许可后方可拿做他用。