Linux系统编程-大作业 

- 组长：1120192092 曾群鸿
- 组员：1120193205 张鹏杰
- 组员：1120190482 李锡汶
- 组员：1820191147 温迪
- 联系方式：871206929@qq.com

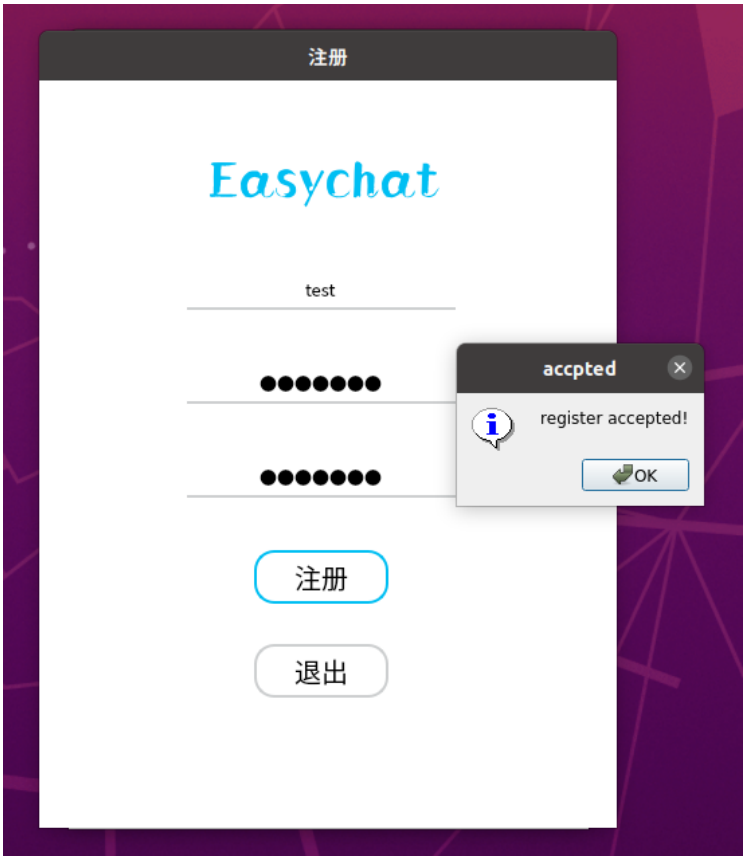
1. 功能介绍 

本次大作业取名为`easyChat`，一个原因是我们希望通过本次实现的软件可以让沟通更加“easy”，令一个原因则是我们采用了较为简洁的设计元素。

对于作业的基本要求，我们在做了更多的细节，如好友上下线的提示，独立的好友申请界面等。在完成基本的作业要求之上，我们还添加了很多有趣的小功能，如头像系统，个性签名，聊天气泡等。

✔注册，登录功能

- 密码强度检查，只有包含数字大小写密码才能通过注册
- 用户注册冲突检测，重复登录冲突检测等

- 

✔主界面

- **头像系统**，随机切换头像，服务器同步更新
- **个性签名**，服务器同步更新
- 合理美观的界面布局

- 

✔好友系统

- 好友界面，**上下线提醒**
- 搜索添加好友
- 收到好友申请界面
- 删除好友功能
-

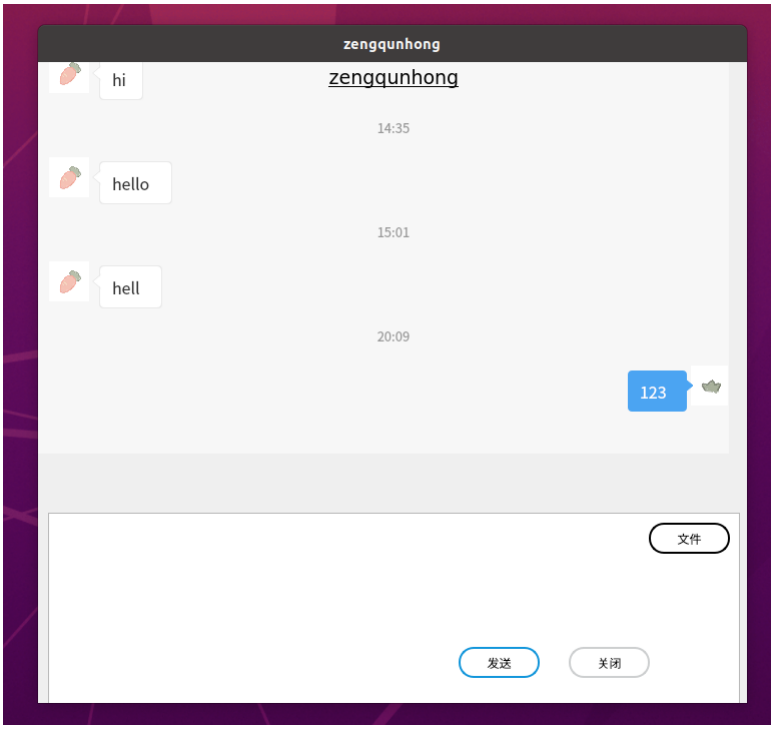

Easychat
zengqunhong
hello world
好友列表
好友申请
进入世界群聊
在线
zhangpengjie
wendi
离线
lixiwen

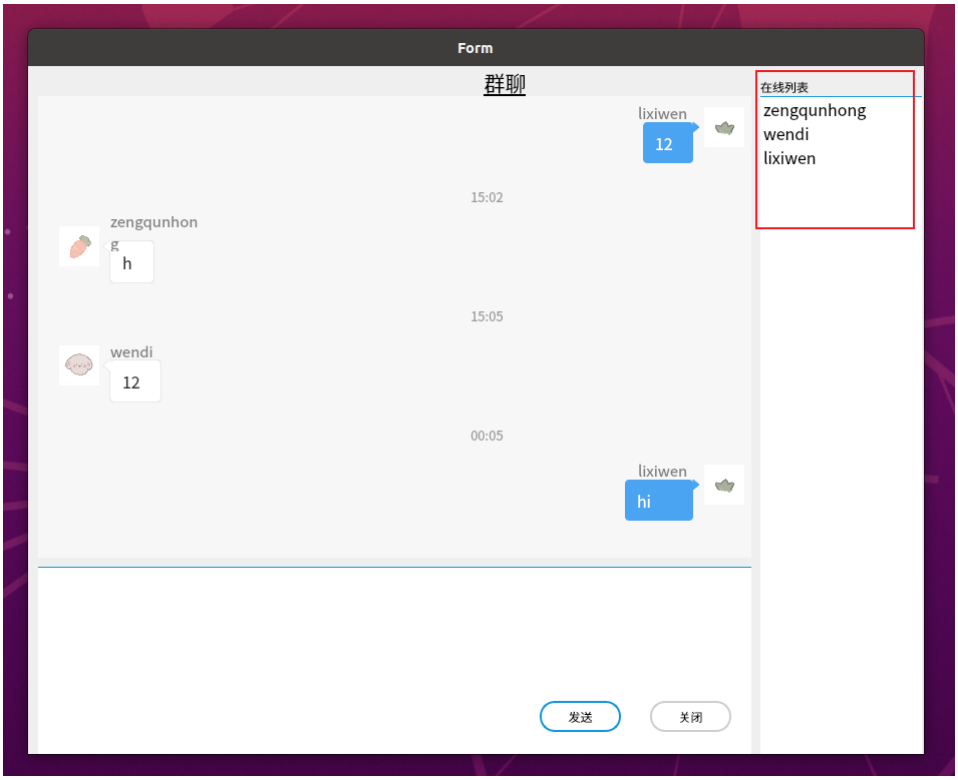
- 
Easychat
zengqunhong
hello world
好友列表
好友申请
添加好友
序号
用户名
时间
操作
添加好友
wendi

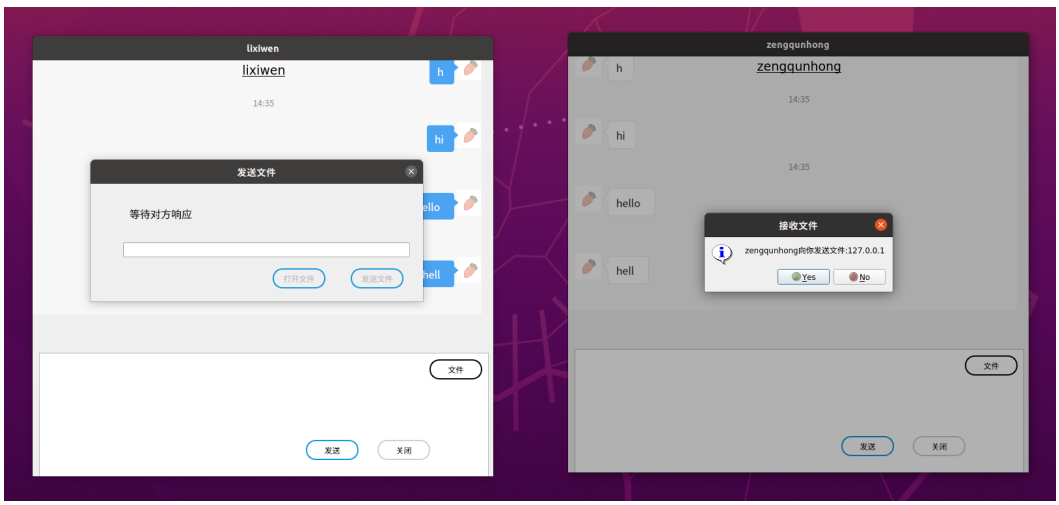
-

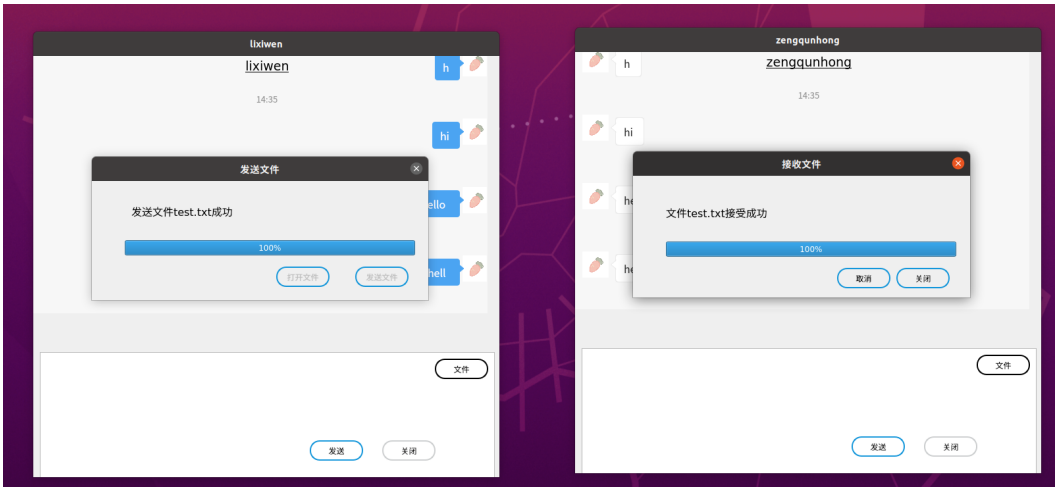
✔聊天系统

- **聊天气泡**
- **通讯内容备份**，打开聊天窗口自动加载聊天记录
- **文件传输**
- 支持**群聊系统**
    - 多人群聊，显示当前聊天室中所有用户
-


zengqunhong
hi
zengqunhong
14:35
hello
15:01
hell
20:09
123
文件
发送
关闭

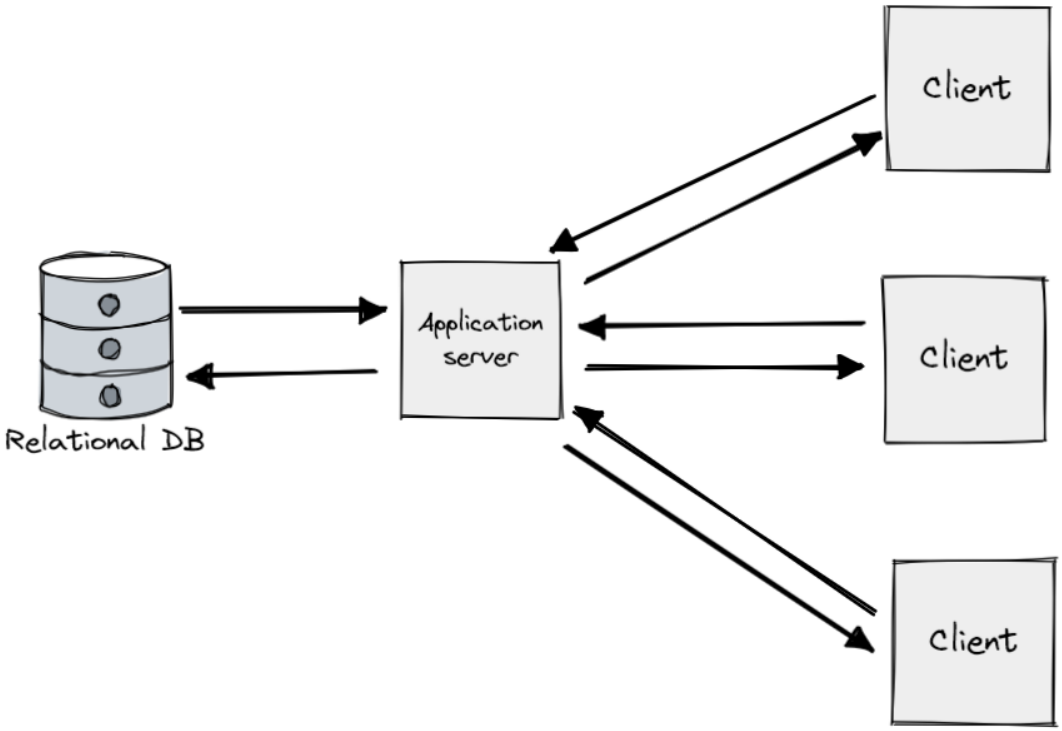
- 
Form
群聊
在线列表
zengqunhong
wendi
lixiwen
lixiwen
12
15:02
zengqunhon
g
h
15:05
wendi
12
00:05
lixiwen
hi
发送
关闭

- 
lixiwen
lixiwen
14:35
h
hi
发送文件
等待对方响应
打开文件
发送文件
文件
发送
关闭
zengqunhong
zengqunhong
14:35
h
hi
14:35
hello
hell
接收文件
zengqunhong向你发送文件:127.0.0.1
Yes
No
文件
发送
关闭

- 
lixiwen
lixiwen
14:35
h
hi
发送文件
发送文件test.txt成功
100%
打开文件
发送文件
文件
发送
关闭
zengqunhong
zengqunhong
14:35
h
hi
接收文件
文件test.txt接受成功
100%
取消
关闭
文件
发送
关闭

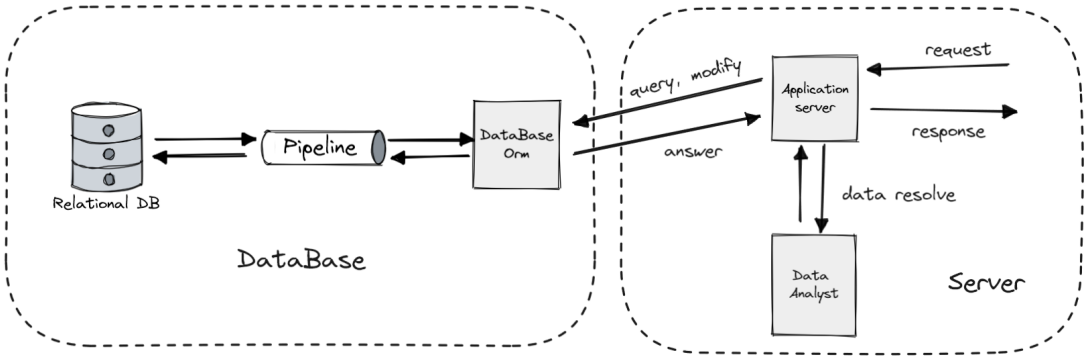
2. 设计方案📌

本作业采用经典的**C/S模型**，分别实现服务器和客户端实现，基本架构如下所示：



2.1 服务端💻

服务器端主要涉及对来自客户端数据报的解析，进行对应的功能解析，同时涉及到数据库的查询修改操作。将服务器端的功能模块进一步细化，我们实现的服务器端架构如下所示：



从左往右分别是数据库，数据处理的一些流水线，为了方便数据库的操作我们仿造`Object Relation Mapping`，将数据库的操作封装成`ORM`框架与服务器进行对接。

服务器端接受来自客户端的各种类型的数据报，为了方便管理这些数据包，我们创建了一个`Data Analyst`来对数据包进行解析，提取数据包的主要内容。服务器根据数据包的内容作出相对应的处理。

在服务器端，我们主要介绍服务器端的后端构成和实现方法：

2.1.1 数据库

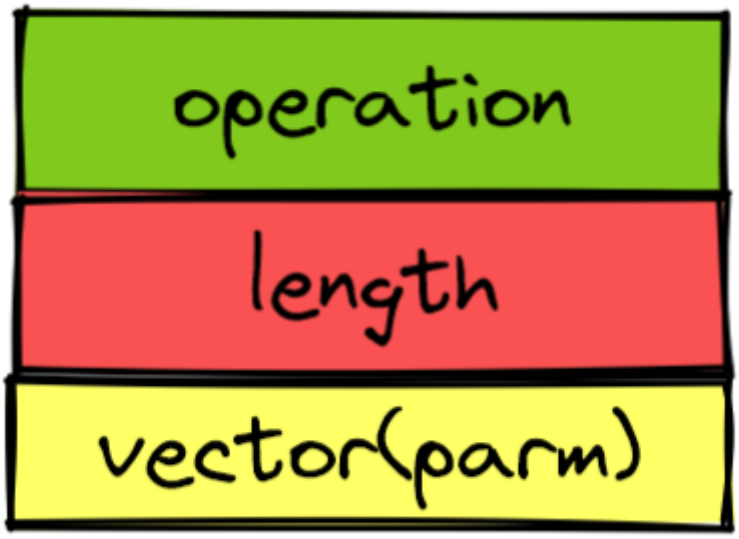
- 代码见`TcpServer/database.h`和`TcpServer/database.cpp`
- 本次作业中一个实现了7张数据库表
    - `user`表，保存用户名和密码，默认为123456
    - `userinfo`表，通过`user`表的主键索引其用户信息，包括用户在应用中的昵称，用户的个性签名，和其他信息（这里用于存取用户的头像对应的图片ID）
    - `relationship`表，把用户之间的关系抽象成一张无向图，该表用于存取用户的好友关系
    - `request`表，用于存取好友申请
    - `chatHistory`表，记录发送者，接受者，发送内容和发送时间，将通讯内容记录下来
    - ...
- 对这些数据库表的基本增删改查，将对应的`SQL`语句封装成类函数，作为“原子”数据库操作方法。
    - 如`registerUser(username, password)`实现在`user`表中插入一行的功能
    - 如`queryFriendById(userID)`实现在`relationship`表中查询好友列表的功能
    - ...

2.1.2 数据库ORM

- 代码见`TcpServer/chatorm.cpp`，`TcpServer/userinfoorm`等
- 将上述数据库类的实例进行封装，提供专一的`object`，并在此基础上对`object`操作进行若干限制
- 一共实现了`chatORM`，`requestORM`，`userORM`，`userinfoORM`等几个相对应的`ORM`类
- 当服务器向数据库查询时，返回一个`ORM`对象给服务器，该对象能够完成相对应数据库表的查询和更新功能
- 通过这一层抽象，让实现服务器的同学不必关心数据库的具体实现，并提供了友好的`C++`类接口，极大提高了作业设计的模块化

2.1.3 数据包解析

- 代码见`TcpServer/dataanalyst.h`和`TcpServer/datapackage.h`
- `datapackage`定义了服务端与客户端之间通信的数据格式，其格式如下所示，其中`operation`表示数据包的类型，`length`用于验证数据包的完整性，后面跟着一个可变长的参数数组
-



- `dataanalyst`定义了数据包的解析工具，其中定义了若干方便的工具函数来实现数据的解析和封装，当服务器或者客户端收到一个数据包，应该使用`dataanalyst`进行解析，然后获取对应的数据操作类型进行操作
- 我们目前支持的通信数据包类型已经非常丰富了，详细见`TcpServer/dataanalyst.cpp`，比较常用的数据包类型有：
    - `message`：用于发送通信消息
    - `login`：用于请求登录
    - `register`：用于请求注册
    - `sendFile`：用于请求发送文件
    - ...

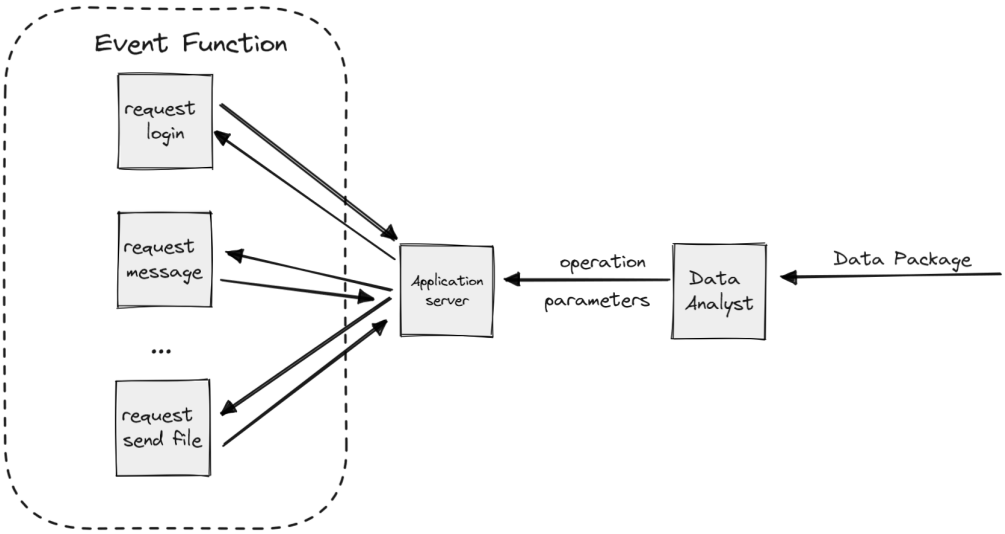
2.1.4 服务端

- 代码见`TcpServer/tcpserver.h`和`TcpServer/tcpserver.cpp`
- 服务器通信主要在`QT`提供的`QTcpServer`上进行封装
- 在服务端上维护了一个`TcpSocket`的列表和一个从用户名到其对应`TcpSocket`的映射表，以实现准确地将某条消息发送给指定的客户端
- 服务器端上实现了一系列的通信方法，如广播机制，将数据包广播给当前在线的所有客户端；如点对点机制，将数据包处理后转发给指定用户；并支持强制将某个用户下线的功能，方便后续的功能扩展
- 除此之外，服务器最重要的功能就是定义了一系列事件处理函数，用于定义当服务器接收到各类数据报时的行为，比较常用的时间处理函数举例：
    - `requestLogin`：客户端请求登录，服务器通过获取数据库`User`表中的元素验证当前传递的数据包是否合法，如果合法服务器检查当前登录用户列表，如果没有登录则把当前登录的用户加入；除此之外，服务器还会

把待登录的用户的好友列表作为数据包传送给客户端，客户端在接受到好友列表后更新UI界面

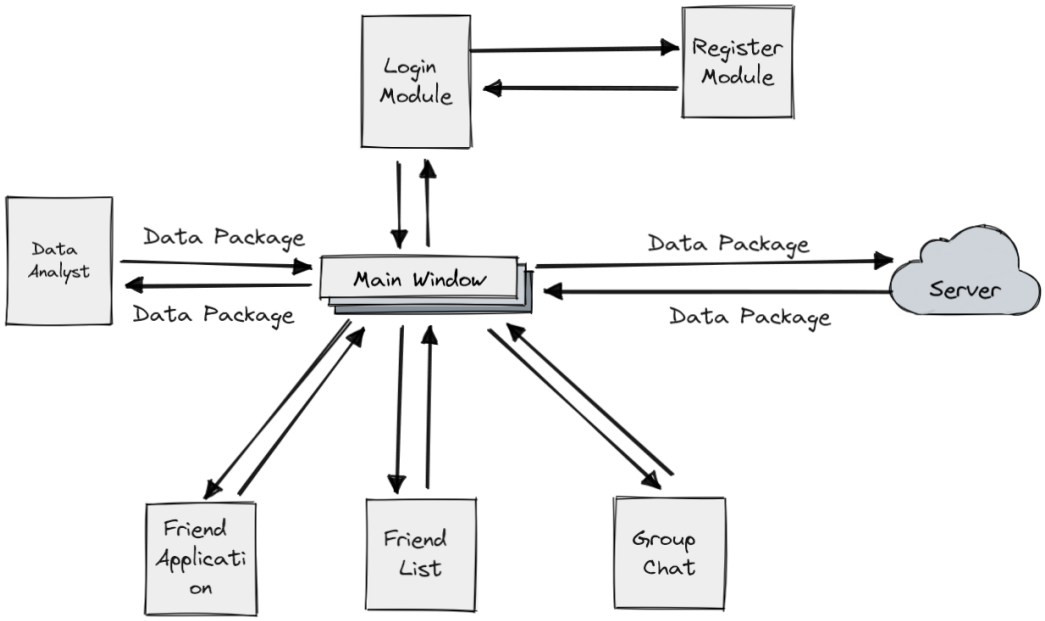
    - `requestSendMessage`：客户端请求发送消息，服务器在接收到该消息后，检查目标用户是否在线，如果在线则使用转发机制发送给目标用户的客户端；如果不在线，则转发，仅仅将消息存入数据库；等用户上线时会从数据库中得到先前其他用户给他发送的消息
    - ...

- 上述描述过程如下图所示：



2.2 客户端💻

客户端相对服务器端，比较复杂的是各种功能的UI界面操作。根据不同功能之间的交互将其相对解耦成几个模块，其架构图如下所示，**我们在客户端中主要介绍各种功能的实现思路**：

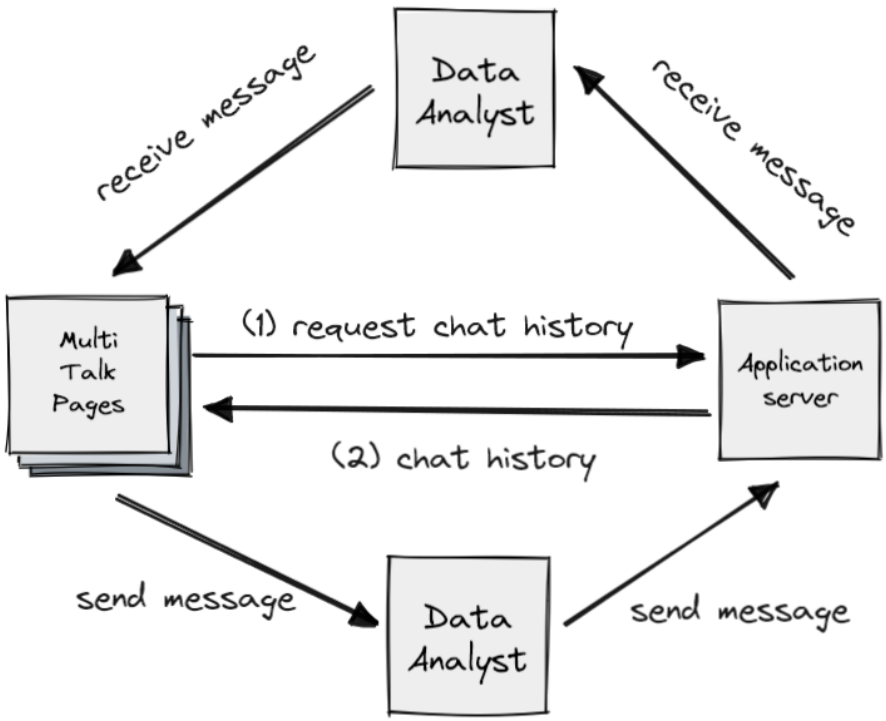


2.2.1 注册、登录功能

- 主要代码框架见`Client/ct3/login.h`和`Client/ct3/register.cpp`
- 注册登录功能作为整个通信软件的入口，实现较为简单，首先利用`QTcpSocket`连接到我们实现的服务器上，与服务器建立相关连接通信，并监听服务器发过来的数据包
- 登录功能实现：在点击登录时，发送上文提到的`login`数据报给服务器，服务器处理后会返回`loginAccepted`或者`loginError`的数据报，根据不同的数据报作出不同的处理即可；如果登录成功，则进入当主界面，与服务器完成后续的通信工作
- 注册功能实现：在点击注册时，调用密码检测模块对密码进行检测，如果密码符合规范，想服务器发送`register`数据报，并根据服务器返回的数据报进行对应的前端动作反馈

2.2.2 群聊和私聊功能

- 代码见`client/ct3/talkpage.h`和`client/ct3/talkpagemany.cpp`
- 群聊和私聊本质上知识广播和转发两种不同通信机制的具体实现，而我们在服务器端预留出了广播和转发的接口用于实现这两种不同的通信功能
- 具体的实现方法为当用户点击发送信息按钮时，根据当前所在页面的不同（私聊还是群聊）向服务器发送不同的数据报，如`message`和`groupMessage`，当服务器收到对应数据包时进行转发或者广播，并保存通信的内容的服务器的数据库中
- **通讯内容备份的实现**：当用户点击进入一个私聊窗口或者群聊窗口时，首先会向服务器发送一个请求聊天记录的数据报请求，服务器根据请求的内容在数据库中查询聊天记录，将聊天记录封装成数据报格式发送给客户端，客户端在收到对应聊天记录时按照前端的UI界面进行显示
-



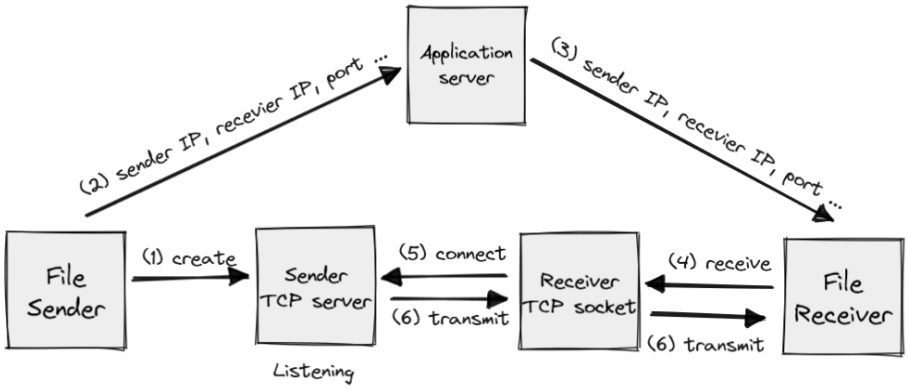
2.2.3 好友系统

- 代码主要见`client/ct3/mainwindow.h`
- 将好友关系看做一张无向图，服务器端在数据库中维护了所有的边关系，并提供了查询接口；当客户端一进入主窗口时，就根据当前用户的信息向服务器请求其好友关系；在获取其好友关系后，在主窗口通过`QT`提供的`Widget`和`Button`接口绘制好友列表，并提供点击功能和右键功能，点击功能弹出私聊窗口，右键功能提供删除好友的选择
- 由于服务器提供了诸多遍历的接口，如判断用户是否在线，所以在传递给客户端数据报时会顺带更新其好友的在线离线状态，客户端前端根据收到的数据报对在线用户和离线用户进行两种不同的显示，在线用户使用纯黑色，而离线用户使用灰色
- 好友申请与接受的实现：同好友列表，根据服务器发送过来的申请数据报，其中有申请的好友名字，申请的时间显示在前端页面中。用户可以点击接受或者拒绝，同样也会有相应的数据包发送到服务器进行处理；当用户点击接受，服务器更新后端数据库，判断好友双方是否在线，向在线的客户端发送`refresh`数据报更新其主界面；当用户点击拒绝，服务器更新后端数据库，将提出的申请作出相应，更新数据状态
- 搜索添加好友的实现：设计对应的前端UI界面，当用户点击时发送数据报到服务器，在`request`数据库表中添加一条记录，将`status`字段设置为待处理；之后当相关好友上线时，会自动查询数据库表，并发送待处理的信息到客户度的好友请求表中

2.2.4 文件传输

- 代码主要见`client/ct3/file_sender.h`和`client/ct3/file_receiver.h`
- 当文件发送方选择发送文件时，创建一个`QTcpServer`并监听指定端口号`8888`，但是这样文件的接受者是无法知道需要连接到哪个服务器的哪个端口进行文件传输的。我们的解决方法是当文件发送方在发送文件时，同时在服务端发送一个数据报

`sendFile`，这个数据报中包含文件的发送者，文件的接受者，发送者的`IP`地址，接受者的`IP`地址，以及传送的文件名。服务端在收到这条数据报后，通知文件的接受者有人想要给他传送文件，并告诉他对方的`IP`地址和端口号，以及文件名。

- 如果文件接受者选择接受，则创建一个`QTcpSocket`连接到发送者的`IP`地址和端口号上，并等待文件的传输
- 文件发送者在发现有新的连接加入后，就向指定的`Socket`发送数据，完成文件的传输
- 

3. 特色与创新✨

- 我们**自定义了网络通信传递的各种数据报格式**，自己**实现了一个服务器系统**，而不是单纯使用`QT`提供的`QTcpSocket`
- 我们遵循**模块化**和**低耦合**的软件工程准则，将一个软件的前端，后端及其各个组成部分相对解耦，实现了一个模块化高可扩展的框架
- 我们实现了一个相对美观的界面，并可以**自定义用户头像和签名**
- 我们支持完善的好友系统，支持**好友搜索**，**好友申请**，**好友删除**，还能根据好友的**在线离线状态给予细节的显示**
- 我们在支持群聊和私聊的基础上，增加了**通讯内容备份的自然展示**（不需要点击查看），**聊天气泡窗口**，并且还有在**群聊界面显示当前聊天室中的所有用户名**

4. 扩展讨论🚀

- 本次作业我们遇到的**最大的困难**是后面改为线上教学以及大三下半学期的考试时间压力，没有充足的时间去完成一个功能丰富的软件。本次作业我们并不是从零实现的，**UI界面和数据库处理部分使用了我们之前写的大三小学期课程设计**；我们在之前的基础上进行分工和扩展，优化了各种基本功能的细节，删去原先不合理的设计，并进行功能扩展，如好友添加，文件传输，群聊，聊天气泡等功能
- 在定义传输数据格式时，其实要考虑的问题应该有很多，我们简单地使用了`$`进行分割，**无法真正做到透明传输**；并且在`DataAnalyst`的设计中不够完善，理论上来说应该使用一个单例类来完成数据的格式解析，并且需要在客户端和服务端保持代码的一致才能保证数据传输的正确性
- 我们在设计服务器的时候**没有考虑到多进程的问题**，由于我们采用的是`C/S`架构，因此服务器理论上会有很大的处理压力；在高并发请求的情况下会导致错误出现，进一步实现应该将服务器的请求和对数据库的操作改为支持多进程，以**保证在大量请求下的正确性和性能**

5. 成员分工 

- 1120192092-曾群鸿：**贡献度**40%，主要完成服务端功能的实现和数据传输实现，实现客户端与服务端的通信对接
- 1120193205-张鹏杰：**贡献度**20%，主要完成数据库的封装，头像，好友列表，好友申请的客户端实现
- 1120190482-李锡汶：**贡献度**20%，主要完成UI界面的系统迁移和修改，登录，私聊，签名的客户端实现
- 1820191147-温迪：**贡献度**20%，主要完成文件传输的实现，群聊和部分好友系统的客户端实现