

本科毕业论文（设计）

基于QT的广域网通讯软件开发

* Development of WAN communication software based on QT

|  |  |
| --- | --- |
| 所 属 学 院 | 数理与电子信息工程学院 |
| 专 业 | 通信工程 |
| 班 级 | 通信194班 |
| 姓 名 | 黄全瑞 |
| 学 号 | 193023020437 |
|  | |
| 指导教师单位 |  |
| 指导教师姓名 | 郭敏杰 |
| 指导教师职称 | 未定级 |
| **完成日期** | **2023年4月10日** |

基于QT的广域网通讯软件开发

**摘要**

为解决Linux桌面操作系统兴起前提下的跨平台网络交流问题，本文设计了一款采用S/C网络架构的跨平台网络通信平台。所设计的通信平台软件在服务器上采用IO复用技术保证了通信平台软件的高并发需求。采用C++ Qt的方式编写UI程序。通过将简单文字通信流程、文件互传等功能集成在所设计的UI界面上，能够轻松解决Windows系统与Linux的互通问题。软件经实测，所设计的系统能够实现预定功能，有利于促进后续互联网和计算机领域的发展。

**关键词：** S/C架构；网络通信；

* Development of WAN communication software based on QT

**Abstract**

In order to solve the cross-platform network communication problem under the rise of Linux desktop operating system, this paper designs a cross-platform network communication platform using S/C network architecture. The designed communication platform software adopts IO multiplexing technology on the server to ensure the high concurrency requirements of the communication platform software. Write UI programs in C++ Qt. By integrating simple text communication processes, file transfer and other functions on the designed UI interface, it can easily solve the problem of interoperability between Windows system and Linux. After actual testing, the designed system can achieve the intended function, which is conducive to promoting the subsequent development of the Internet and computer fields.

**Key words:** S/C architecture; Network communication;

目录

[1 绪论 1](#_Toc134008019)

[1.1 研究背景及意义 1](#_Toc134008020)

[1.2 研究内容以及难点 1](#_Toc134008021)

[2 研究思路和方案设计 3](#_Toc134008022)

[2.1 项目简介 3](#_Toc134008023)

[2.2 主流实现的基本通信功能 3](#_Toc134008024)

[2.3 程序通信架构方案 4](#_Toc134008025)

[2.4 界面程序架构方案 5](#_Toc134008026)

[2.5 技术需求分析 5](#_Toc134008027)

[3 技术探讨与选择 6](#_Toc134008028)

[3.1 探讨内容概要 6](#_Toc134008029)

[3.2 网络协议 6](#_Toc134008030)

[3.2.1 网络协议简介 6](#_Toc134008031)

[3.2.2 TCP协议 6](#_Toc134008032)

[3.2.3 UPD协议 6](#_Toc134008033)

[3.2.4 HTTP协议 7](#_Toc134008034)

[3.2.5 WebSockect协议 7](#_Toc134008035)

[3.2.6 协议选择 7](#_Toc134008036)

[3.3 高并发技术 8](#_Toc134008037)

[3.3.1 高并发服务器开发简介 8](#_Toc134008038)

[3.3.2 select技术 8](#_Toc134008039)

[3.3.3 poll技术 8](#_Toc134008040)

[3.3.4 epoll技术 9](#_Toc134008041)

[3.4 数据库 9](#_Toc134008042)

[3.4.1 数据库简介 9](#_Toc134008043)

[3.4.2 数据库选择 9](#_Toc134008044)

[3.5 网络开源库 10](#_Toc134008045)

[3.5.1 为什么选择开源库 10](#_Toc134008046)

[3.5.2 网络编程的常见问题 10](#_Toc134008047)

[3.5.3 网络开源库选择 11](#_Toc134008048)

[3.6 图形框架 11](#_Toc134008049)

[3.6.1 图形框架简介 11](#_Toc134008050)

[3.6.2 图形框架选择 12](#_Toc134008051)

[4 跨平台软件设计 13](#_Toc134008052)

[4.1 章节概要 13](#_Toc134008053)

[4.2 客户端架构设计 13](#_Toc134008054)

[4.3 登录流程 14](#_Toc134008055)

[4.4 登录状态切换 15](#_Toc134008056)

[4.5 数据交换流程 16](#_Toc134008057)

[4.6 数据交换状态 17](#_Toc134008058)

[4.7 服务器架构设计 18](#_Toc134008059)

[4.8 注册请求处理 19](#_Toc134008060)

[4.9 登录请求处理 20](#_Toc134008061)

[4.10 转发请求处理 21](#_Toc134008062)

[5 交换协议与数据模型 22](#_Toc134008063)

[5.1 章节概要 22](#_Toc134008064)

[5.2 请求响应模型 22](#_Toc134008065)

[5.3 数据转发模型 23](#_Toc134008066)

[5.4 交互协议结构 24](#_Toc134008067)

[6 实测结果与讨论 25](#_Toc134008068)

[6.1 章节概要 25](#_Toc134008069)

[6.2 登录窗口测试 25](#_Toc134008070)

[6.3 聊天窗口测试 26](#_Toc134008071)

[6.4 文件进度测试 27](#_Toc134008072)

[6.5 好友申请测试 28](#_Toc134008073)

[7 总结和展望 30](#_Toc134008074)

[8 参考文献 31](#_Toc134008075)

[9 致谢 32](#_Toc134008076)

[10 附录 A 完整的转发协议文件 33](#_Toc134008077)

[11 附录 B 服务器任务分发代码 38](#_Toc134008078)

[12 附录 C 客户端的任务转发代码 40](#_Toc134008079)

# 绪论

## 研究背景及意义

网络聊天工具的发展依托于现代互联网的流行，其在一定程度上取代了过去的异地通话及短信服务。作为一种基于计算机网络的终端通信工具，网络聊天工具最显著的特点便是便捷性与灵活性。现阶段最具流行性的便是腾讯公司旗下的QQ和微信两款通信软件。当然其中也包括诸如探探、陌陌等成人向交友软件。虽然此类软件在社交方式与主流业务上有很多不同，但其本质仍旧是社交通信软件。

但现阶段主流的网络聊天工具主要服务于Windows操作系统平台，随着Linux桌面操作系统的兴起，已经有一部分用于开始选择使用Linux系统作为日常使用的操作系统。但由于Linux系统主要服务于开发者而非普通用户，导致Linux操作系统在日常用户的软件生态上非常缺失，在Linux操作系统的上可用的聊天软件可谓抓襟见肘。为解决这样的问题，本文拟研究建立一个可跨平台的、既支持以软件的形式启动客户端、还支持以浏览器网页形式启动客户端聊天通信软件，并以达到快捷、即时的通信目的。

基于以上论述可以看出，对于网络聊天工具的研究在当今社会具有一定的重要性，它有效的提高了沟通效率，缩短了交流所需要的时间成本，在个人生活的各方面都起着重要作用。同时，网络聊天工具也为企业提供了一个方便的沟通渠道，有助于公司提高客户服务效率。再者，网络聊天工具也可以为学校提供一种有效的学习沟通方式。因为有了便捷的网络聊天工具，它们极大地改善社会的沟通方式以及沟通效率。

## 研究内容以及难点

本文拟基于Qt图形框架开发的广域网通信工具。所设计的软件将在Qt、C++和Linux环境下开发，并实现注册、登录、信息发送、文件传输等功能。

本文的研究难点在于，在高并发、多连接的高性能服务器网络编程上很难突破现有框架限制，因为一个高质量的网络框架需要经过工业级别的多年锤炼才可以流行于市面上。同时，采用WebSocket协议作为底层协议虽然为在未来实现浏览器客户端留下了可能，但应用层协议带来的开销往往是性能的瓶颈，很难突破。再者，在服务器上，数据库交互也是造成性能瓶颈的主要原因，所以对于数据表进行范式设计与优化、对数据的类型结构拆解也是难点之一。

# 研究思路和方案设计

## 项目简介

本项目是主要实现一个聊天软件，聊天软件的业务一般为文字聊天、语音/视频聊天、文件传输、多开同时会话、可发送表情及动画、保存聊天记录、创建群组等七大功能。而本项目将重心放置到UI结构设计、软件架构、传输交换协议处理、以及用户交互上来，业务逻辑部分暂不讨论。

该项目只提供了登录与账号注册功能、发送文字与发送表情包功能、传输文件功能、添加好友功能、保存聊天记录功能，且将服务器部署至互联网。

## 主流实现的基本通信功能

该软件依照现在比较流行的聊天软件的主流逻辑与通信功能做出来以下几点的设计，分别是：

1. 提供互联网通信，部署公网服务器，采用S/C网络架构通信。
2. 提供账号注册功能，用于登陆验证用户身份。
3. 提供好友申请等组群关系。
4. 提供基本的文字消息发送功能。
5. 提供基本的表情包图片发送功能。
6. 提供基本的文件传输功能。
7. 提供历史消息记录功能。

依据上述的功能，接下来将对程序的总体框架进行设计，对整体程序的通信框架，以及UI界面的总体布局提出合理的架构设计方案。

## 程序通信架构方案

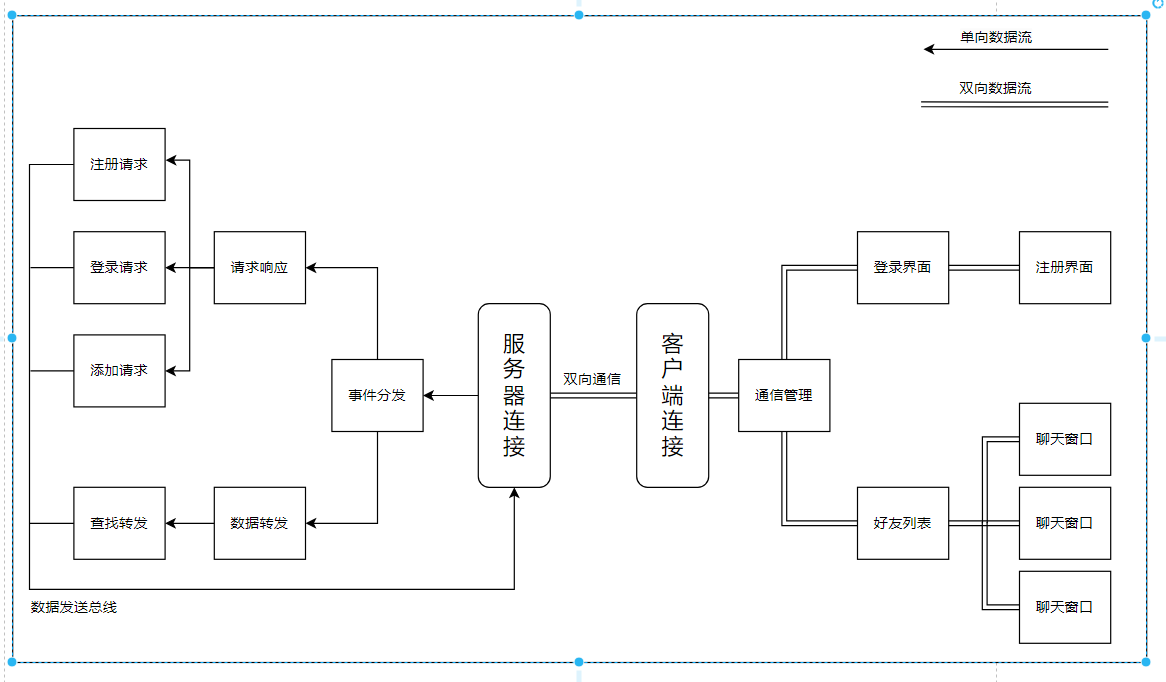


图 2.1 通信架构

由图 2.1 通信架构得知程序的整体通信架构，程序的通信架构分为两个部分，分别是客户端连接与服务器连接。从图中可以看出，客户端采用双向的数据流，但被层级限制，数据需要层层传递才能到达最深处，这样的设计有利于模块化程序的结构。所有数据必须经过通信管理层，使得客户端功能模块开发不需要关心网络通信，达到模块解耦的功能，有利于业务功能的开发。

服务器的数据流则被设置为单向流数据传输，这样设计的原因是因为任务需要被事件分发器分发到指定的任务处理函数。处理模块有直接与客户端通信的权利，当任务处理函数执行完成之后，任务处理函数直接与客户端通信，对请求任务进行反馈。

## 界面程序架构方案

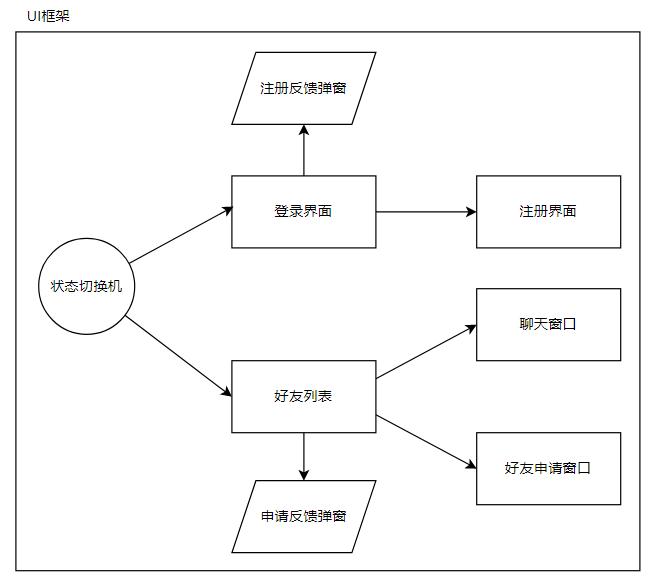


图 2.2 界面架构

由图 2.2 界面架构得知程序图形界面的总体架构，图形界面架构被分为两个部分，分别是登录前的登录界面和登录之后的好友列表，两个UI界面互不干扰，由状态切换机来控制两者的显示。

登录界面分别控制注册界面与注册反馈弹窗，注册功能的界面由登录界面启动。好友列表则控制聊天窗口、好友申请窗口、申请反馈弹窗，聊天功能与好友申请由好友列表窗口提供。

## 技术需求分析

本项目需要实现高性能网络服务器，实现UI图形界面与用于的交互，实现客户端与服务器的数据交互，需要数据库存储用于信息，还涉及到Windows与Linux两个操作系统的跨平台问题。从开发者角度出发，该软件开发触及到网络服务器开发，桌面UI图形界面开发，跨平台开发，数据库操作，应用层业务协议设计，网络服务器部署等技术需求，对技术的深度与广度都有一定的需求。下一章将对技术需求做详细探讨。

# 技术探讨与选择

## 探讨内容概要

技术探讨主要是用于了解开发本项目需要使用到的各类技术，因为该项目的开发涉及到的技术栈非常多，且各类技术难度在短时间内难以掌握。通过该章节对各类技术的简介，以及各类技术的优缺点选择之后，可以对本项目需要使用到的技术栈有一个大概的了解。

## 网络协议

### 网络协议简介

既然要实现网络聊天功能，网络间的通信协议便是不可不谈的话题，网络协议是计算机之间通信的桥梁。在目前的主流的网络协议分别为TCP协议、UPD协议、HTTP协议、WebSockect协议。其中TCP协议和UPD协议是传输层的网络协议，应用范围非常广泛，大部分的通信软件所使用的网络协议，为其中一种很或者两者皆有。HTTP协议和WebSockect协议则是建立在TCP协议的应用层通信协议，主要用于服务器与浏览器的通信。接下里将对比各个协议之间的优点与劣势。

### TCP协议

TCP提供的面向连接、提供可靠交付、面向字节流、稳定且安全可靠协议解决了应用层的零丢失问题，所以TCP协议才会被大量使用。尽管TCP协议需要三次握手建立连接，四次挥手的操作，而且是面向字节流，在应用层开发上会存在粘包以及压栈等问题，不过凭借着在传输层的滑动窗口的拥塞控制、全双工的通信连接减少了应用层对于网络流量的管理，而且使用连续ARQ的重传机制提速，速度并不慢，由此看来TCP协议被广泛采用不无道理。

### UPD协议

UDP提供了无连接、尽最大努力交付、面向报文、是快速且不安全功能，其特点就是快速，且是无连接无反馈的快速发送，但缺点也是无连接造成的保报文丢失等不可靠因素，需要应用层实现可靠通信。在网络环境良好的区域上，建立在可靠的UDP协议基础之上的应用协议将得到非常优秀交换体验，而当网络不稳定时，应用层实现的可靠和重发功能往往会将UDP的快速优势清扫的一干二净，速度比TCP还低数倍。

### HTTP协议

HTTP协议最开始是出现在浏览器上的一种基于TCP的应用层协议，该协议的主要任务就是用浏览器向数据服务器发送请求报文，服务器根据报发内容反馈内容并马上断开连接。所以HTTP协议的特点就是简单快速、灵活且无连接、无状态的一种短时TCP连接。其中无连接无状态指的是HTTP建立了请求/响应模型来对服务器进行通信，一旦本次内容提交并由服务器反馈完成则不保留如何连接记录。浏览器想保存信息就必须将全部的内容重新发送，而这就是浏览器会使用Cookies记录的原因，Cookies只是保证了客户端的信息记录，对于服务器来说依旧是一次全新的对话。

### WebSockect协议

WebSockect协议是一种基于TCP的应用层协议，与HTTP协议一样用于服务器与浏览器的通信。WebSockect协议主要解决了HTTP协议推送困难的问题。HTTP协议想要实现推送功能，所用的技术是短轮询。轮询是指由浏览器每隔一段时间向服务器发出 HTTP 请求，然后服务器将新数据到浏览器。之所以由浏览器定时请求服务器的原因就在于HTTP协议是短连接、无状态的，服务器无法主动向浏览器发送内容，只能被动回应。为了解决这个问题WebSockect协议采用了长连接有状态的连接方式，保留浏览器信息，使得服务器可以主动的向浏览器推送消息。

### 协议选择

本项目选择WebSockect协议作为主要通信协议的原因很简单，因为WebSockect协议可以由服务器主动向浏览器推向消息，且协议包头只有较小的开销，支持二进制内容的传输，为文件传输留下方便，内容是带传输格式，采用WebSockect协议使得网络部分的开发相对简单。

## 高并发技术

### 高并发服务器开发简介

高并发指的是服务器在短时间内遇到大量操作请求的情况，高并发服务器开发则尽最大可能的处理更多的请求。在实现通信软件之前，必须考虑到服务器通信的承载量，保证客户端的高并发需要，以及最大可能的发挥出服务器性能。

服务器早起采用的并发技术为多进程服务器与多线程服务器。多进程服务器指的是服务器一旦建立新连接则开启一个新进程与客户端连接，并完成通信，主进程继续监听。多线程服务器的连接处理类似，只不过由建立新进程改为了建立新线程，但不管是建立进程还是线程，对于服务器的资源消耗都相当高，于是就引出了IO复用技术。

IO复用技术，即用一个线程处理大量的IO操作，可以减少服务器的资源消耗，让服务器可以承载更多的IO连接与处理请求。主流的IO复用技术有三种，分别是select技术、poll技术和epoll技术，接下来将分别介绍这三种技术。

### select技术

select技术是将已经连接的套接字放入select连接集合进行管理， select将轮询整个连接集合，即连接集合中的套接字主动进行轮询监测，一旦发现连接的套接字可写或可读则对套接字进行标记，提示开发者套接字可操作。

select技术存在一个缺点，即连接数不能超过1024，也就是说最大只能接收1024个套接字，其原因是因为连接数量增大会影响性能。

### poll技术

poll技术与select技术十分类型，是对select技术的优化，提供了无限制的套接字连接数量，但连接数的增加依旧会影响性能，原因是poll技术与select技术一样，采用无差别轮询，即只有轮询到套接字才知道该套接字是否可读或者可写。

采用无差别轮询原因是因为知道某一个套接字可读或者可写，但不知道具体是哪一个套接字。两者都在不活跃的套接字上消耗了太多的时间，从而导致连接数影响了性能。不活跃指的是套接字不可读，活跃指套接字可读，且数据已经送往内核缓冲区。

### epoll技术

epoll技术是与select技术、poll技术相同的是，依靠连接集合管理大量已经连接的套接字，并进行监控。不同的是，通知方式不再是轮询而是改为事件驱动，只有活跃的套接字才会加入事件循环，使得epoll的事件循环中的每一个轮询都有事件可以处理，其中包括可读、可写、连接、关闭等事件。epoll技术不会被不活跃的套接字影响性能，使得无限制连接成为可能，让服务器轻易可处理高达百万的连接请求。

技术选择

介绍了IO复用技术select、poll和epoll的特点之后，本次项目选择了epoll技术作为服务器的高并发解决方案，因为其独特的机制可以提供更高的连接数量，以及减少服务器的性能开销。尽管在小型项目中，三者都可以轻易完成对服务器部署的需求，但选择epoll技术则表现出了对于未来前瞻性以及期盼。

## 数据库

### 数据库简介

数据库是指以一定方式储存在一起、能与多个用户共享、具有尽可能小的冗余度、与应用程序彼此独立的数据集合。数据库的主要用于存储大量的数据信息，且与应用程序实现分离，并能快速查询数据和操作数据。

主流的数据库有关系型数据库MySql、Oracle、SqlServer、Sqlist3、Access等，其中MySql、Oracle、SqlServer数据库需要与数据库服务器进行连接，此类数据库用于大规模数据或超大规模数据管理与存储。Sqlist、Access数据库则是轻量级，用于存储小量数据，甚至是文本级别的微量数据。

### 数据库选择

该项目对数据库的选择主要用于服务器的存储，考虑到服务器的部署问题与服务器数据量较小等因素，最后选择Sqlist3数据库主要数据存储。选择Sqlist3数据库的理由很简单，Sqlist3数据库无需连接服务器，产生的db数据库文件可以随时迁移，且因为受到广泛使用已经有一套完善的API可用于开发，查询速度与MySql媲美，非常适合小型项目作为主选数据库。

## 网络开源库

### 为什么选择开源库

网络编程绝不是套接字的发送与接收这么简单，服务器开发有一套完善的规则，其中开源库主要是为网络编程搭建一个框架，搭建一个高性能服务器程序框架，一个高性能服务器离不开高性能IO、IO复用、定时器、多线程编程、线程池等必要组件，以及协议解析、网络数据传输、解封包等数据协议的封装与解析。以及网络编程存在的问题，接下来将列举出网络编程的常见问题。

### 网络编程的常见问题

1. TCP粘包问题，非常常见，因为是TCP是面向流字节传输的，数据无边界，如果对数据进行封装，会导致收发数据的结构不对等。
2. UDP报文丢失问题，使用UDP是必须在应用层实现可靠的重发策略，保证数据可靠或者选择性丢弃。
3. 超时重传问题，因为网络复杂，偶尔会出现延时导致数据无法及时发送或者接收，导致超时，需要在超时后选择断开连接或者重新传输数据。
4. TCP长连接问题，虽然TCP底层已经存在保活设置，不过要是由于硬件设备的突然中断所导致的连接断开则不反馈，所以应有层需要实现自己的心跳（或称ping-pong）机制来保持长连接与处理意外断开。
5. 流量控制问题，虽然TCP提供来滑动窗口来做拥塞控制，但是确是基于路由器交换的，如果如果是上层应用的不当操作所造成的TCP数据压栈会影响其他进程的无法发送数据，所以要对数据发送的速度做限制，以避免应用层的数据大量发送，达到流量控制的目的。
6. 断线重连问题,TCP是一个长连接的协议，但是网络却不是永远都这么好，当网络发生错误导致TCP套接字断开时应用层应该考虑TCP的重新连接以达到用户友好的目的。
7. 通信加密问题，TCP采用的是明文传输，在数据安全性上未留下余地，导致重要信息被轻易获取，所以在互联网中传输数据时安全性不得不好好考虑一番。
8. 大小端问题，由于硬件的不统一，在内存的硬件存放上存在大端、小端两种不同的内存数据正反拜访顺序，它们的数据在内存中按字节端相反，如果没有对大小端进行处理则会读取到与预期不符的数据。
9. 多线程顺序问题，在客户端中发送消息通常只有单线程发送与接收，但在服务器上由于到同时处理多个sock连接的数据，所有都会提前生成线程池等待数据处理，而多线程在发送或者接收时可能会因为时间并发，如果读取的返回时间不一致会导致先读的后返回，从而导致获取的内容顺序错乱。
10. 串包问题，该问题是由原本发送请求的连接因为网络原因断开之后，其原本的sock被新连接客户端占用，而服务器接收到请求并进行反馈时却将反馈内容返送到了占用的sock上，导致断线重连的客户端未能接收到反馈包，而服务器错误的将包发送到被占用的sock上引发的错误。

### 网络开源库选择

C++语言是一个庞大且复杂的语言，实现各种特性与标准库，却迟迟没有在标准库中添加网络库，其原因是各领域对网络库的需求不一致，很难达到统一。尽管如此C++的能用的网络库却非常多，其原因是C++开发者喜欢造轮子。目前主流的网络库有libevent、libev、libuv、libhv、ACE、libcurl、boost::asio等众多稳定可靠的网络库，且实现各类通信协议以及封装跨平台接口。

本次选择的网络库是libhv网络开源库，该库实现了WebSocket协议，封装跨平台C++API，实现SSL数据传输安全，接口简单易用，性能很高，且开源代码可自定义。对于无特殊需求的服务器开发，上述任选一种网络框架也能达到需求。

## 图形框架

### 图形框架简介

开发桌面UI的图形框架有很多，其中C# 语言有 WinForm、WPF等开发框架，C++ 有Qt、MFC、WTL、wxWidgets等多个图形界面可供选择，Java 语言 有AWT、Swing两大框架，在图像界面框架上选择众多。三个编程语言中C#是最适合开发界面的，但对Linux系统跨平台的支持并不友好，故不做考虑。

### 图形框架选择

考虑到本项目需要跨平台开发，且自身对C++编程语言更加熟悉，所以选择了C++ and Qt的图形框架进行UI界面的开发。选择Qt的一个重要原因是，Qt提供了完善的API开发帮助文档，对于开发学习非常有帮助，且对跨平台支持非常友好。

# 跨平台软件设计

## 章节概要

本章将列出客户端以及服务器的实现细节，对整个程序的实现做出一个整体的展示。其中将详细介绍客户端与服务器的设计框架，以及解析程序运行流程图的实现逻辑与设计思路。

## 客户端架构设计

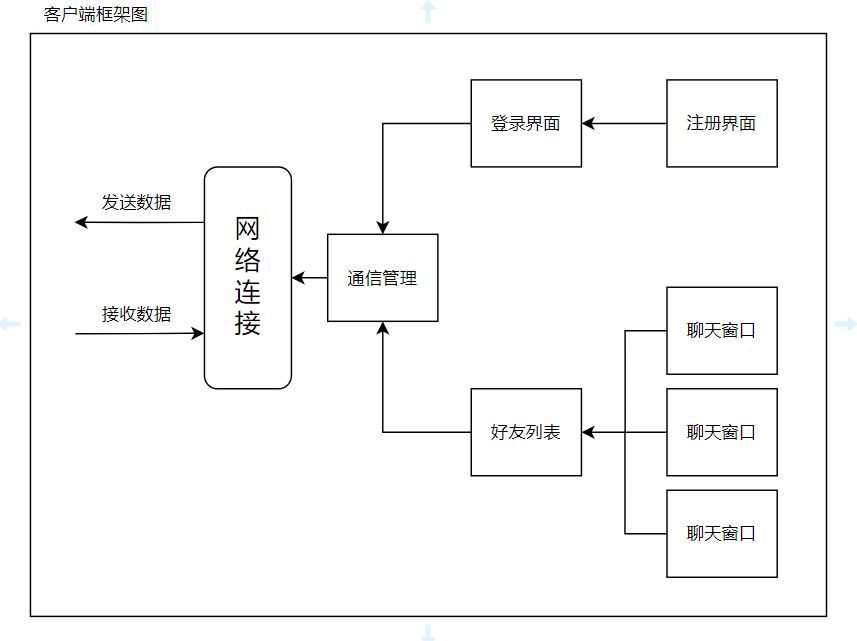


图 4.1 客户端架构

由图 4.1 客户端架构可以看出，客户端的框架设计中在网络接连处安插了一层通信管理，通信管理负责整个客户端程序与服务器的数据交互，所有从客户端发送或者从服务器接收的消息都用通信管理层统一分配到各个模块。

其中登录模块与好友列表模块是分离的，由通信管理模块进行整合，如登录成功时，登录模块以及完成任务，则由通信管理模块负责将其关闭，再启动好友列表模块。

## 登录流程

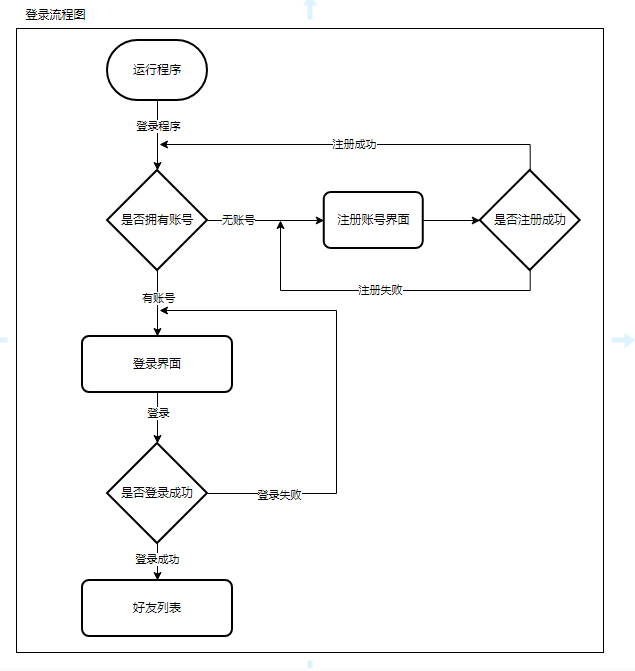


图 4.2 登录流程

从图 4.2 登录流程中可以看出，程序的登录流程。程序的登录大致可以分为两个状态，有账号与无账号状态。当有账号时，可直接输入账号密码登录。无账号时，需申请账号，并登录到好友列表界面。

## 登录状态切换

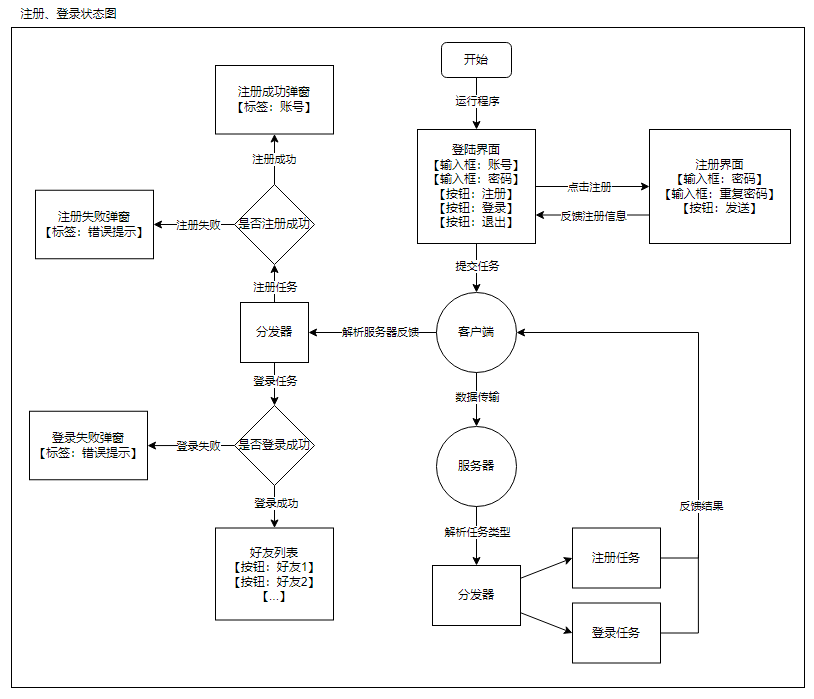


图 4.3 登录状态

从图 4.3 登录状态中可以看出，客户端登录、注册时与数据与服务器的交互状态，以及UI切换的状态。图中可以清晰的看出，程序运行到成功登录的所有状态。首先程序运行会弹出登录界面，登录界面提供了UI控件，其中包括账号密码输入，以及登录、注册等按钮。

如果用户进入注册界面并注册账号时，注册界面会收集用户的信息，并将注册信息回传到登录界面，由登录界面程序将注册信息发送到服务器。服务器通过任务分发器将任务分发到指定的处理函数，再由处理函数将处理结果反馈到客户端。客户端通过任务反馈分发器做出不同状态的切换。

客户端上的任务反馈分发器将判断注册、登录是否成功，并执行指定任务。如果注册成功则显示注册的账号，失败则反馈错误信息。

当用户登录成功时，好友列表UI将显示，注册、登录UI将被关闭。

## 数据交换流程

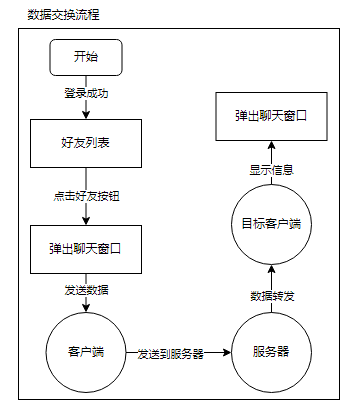


图 4.4 数据交换流程

从图 4.4 数据交换流程中可以看出，程序在从一方客户端将数据发送到目标客户端时的基本流程。当用户点击好友按钮时，会弹出对应的聊天窗口，此时可以与之通信。所有发送的数据由服务器转发到目标客户端上，发送的数据将在目标客户端中对应的发送方的聊天窗口上显示。

## 数据交换状态

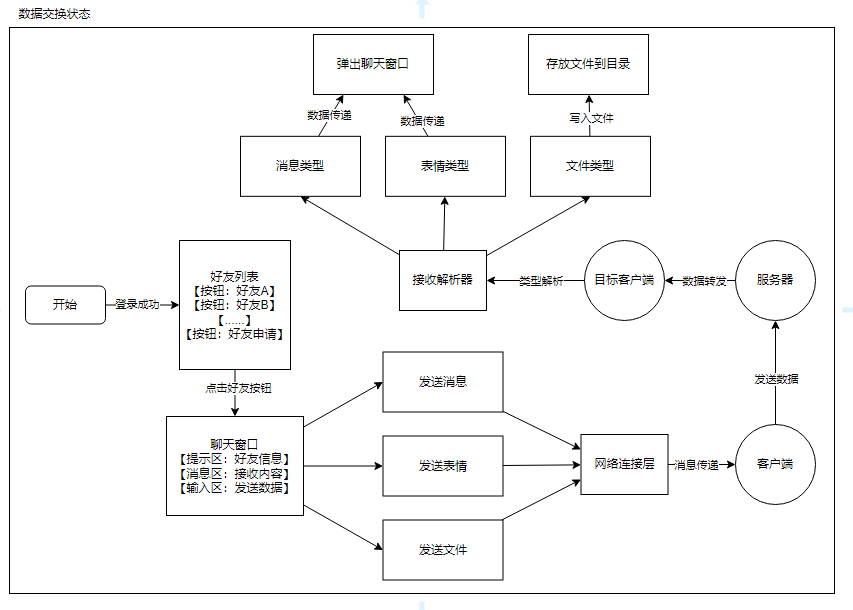


图 4.5 数据交换状态

由图 4.5 数据交换状态得知，发送数据交换时状态响应，UI界面的状态反应。数据交换主要有聊天窗口提供，聊天窗口由数据发送和接收数据两大部件组成。数据发送主要提供三大功能，分别是消息、表情包、文件。

发送任何一种类型的数据，都首先传递到网络连接层，将消息类型预处理，之后发送到服务器。服务器将数据转发到目标客户端时，由接收解析器分析类型，再调用对应的处理函数。不管收到何种消息类型，目标客户端都会将聊天窗口弹出，以表示接收到对话。

## 服务器架构设计

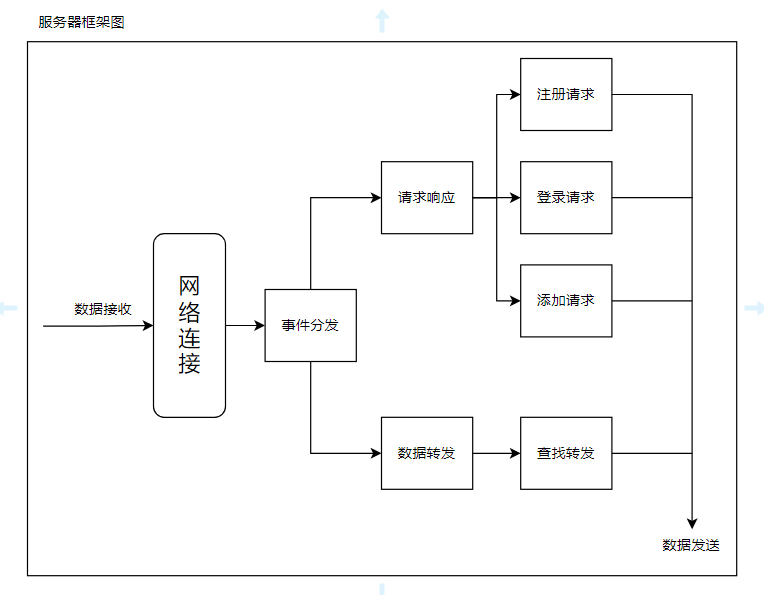


图 4.6 服务器架构

由图 4.6 服务器架构可以得知，当客户端向服务器发送消息时，由事件分发器判断类型，如果是数据转发事件则交由数据转发模块进行处理。如果是请求响应事件则再一步确认任务类型，最后提交到具体的任务处理函数中。可以看出服务器的数据发送并不经过网络连接部分，而是直接发出，这意味着每个任务处理函数都可以拿到直接与客户端通信的权限。这一点与客户端的通行方式不同。

## 注册请求处理

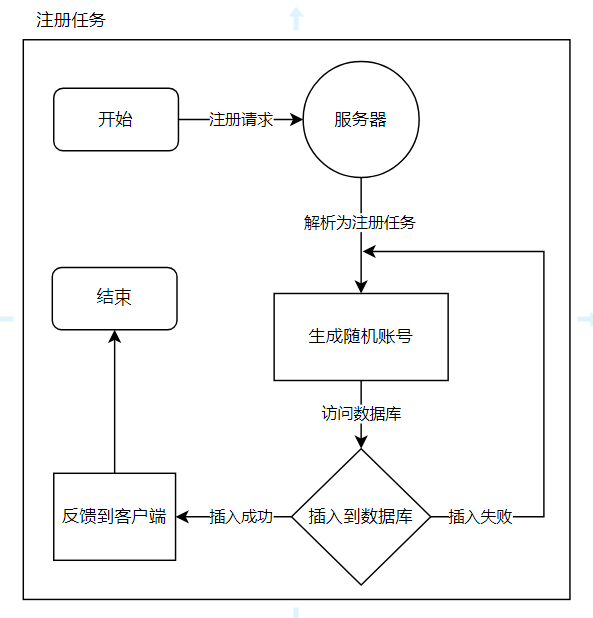


图 4.7 注册任务

从图 4.7 注册任务可知，服务器在处理客户端发起注册任务时的响应流程。服务器首先分析任务类型，得出为注册类型之后生成10位随机的正整数账号，并试图插入数据库。数据库的账号字段被设置成主键，不可重复插入，如果插入失败则代表生成的账号为重复值，重新生成并尝试插入。插入成功则将账号信息反馈到发起请求任务的客户端。

## 登录请求处理

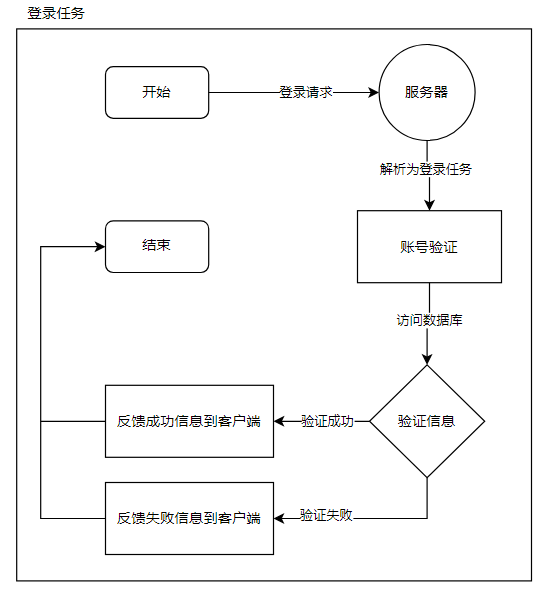


图 4.8 登录任务

由图 4.8 登录任务可看出服务器处理登录请求时的基本流程。服务器解析出登录任务后开始验证登录信息，登录信息存放在数据库中，通过查询数据库对比验证，判断客户端的登录信息是否与数据库内的信息一致。如果一致则将登录成功的内容反馈到客户端，否则反馈失败。

## 转发请求处理

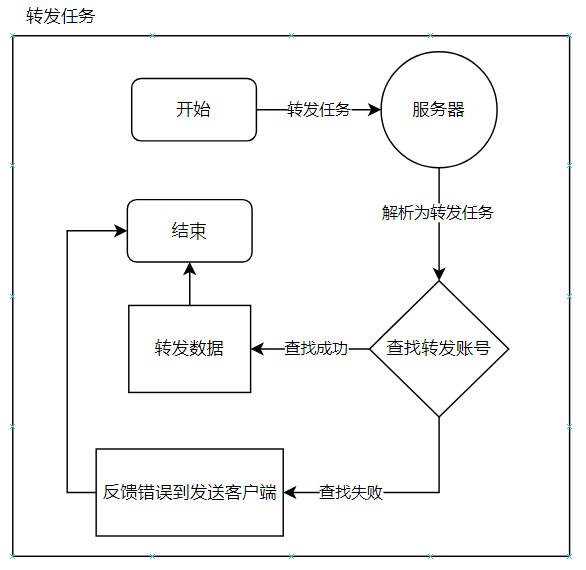


图 4.9 转发任务

由图 4.9 转发任务可知服务器处理转发任务的流程。当服务器解析出任务类型为转发任务时，调用转发处理函数，该函数通过转发目标账号，查询该账号是否登录，如果登录则转发到目标客户端。当目标账号未登录时，将返回提示信息到发送请求的客户端。

服务器在所有账号登录成功时，将登录账号与套接字绑定到登录索引容器，当服务器需要转发任务时，登录索引容器通过Key-Value的形式快速通过账号索引到套接字，此时将需要转发的数据写入该套接字即可完成转发。

# 交换协议与数据模型

## 章节概要

本章主要讲解程序设计时数据交互的业务协议，以及业务协议对功能实现的影响。在网络程序中，所有的数据都需要遵循一定的协议才能被解读，新功能的制定也需要协议支持。应用层的业务协议是服务器与客户端之间的桥梁，通过协议搭建起一套完整的通信机制。

本程序使用了两个协议模型，分别是请求响应模式与数据转发模型，基于这两个模型已经实现了账号登录、发送消息、文件传输等一系列功能。接下来将接收功能的协议模型与功能的实现细节。

## 请求响应模型

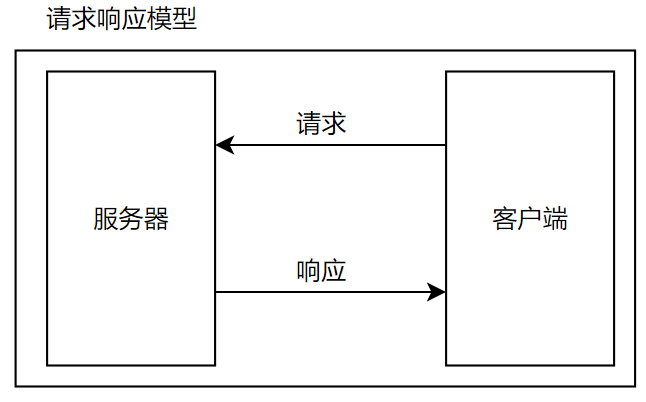


图 5.1请求响应模型

从图 5.1请求响应模型可以看出，客户单与服务器的数据传递方式非常的简单，是一问一答的形式。当客户端向服务器发送请求时，服务器会处理客户端的请求，并客户端发送响应包。这种形式的协议模型最常见的便是Web服务器与浏览器的信息交互，对功能的处理十分的简单有效，在注册、登录、退出等功能我们引用了这种通信形式。

## 数据转发模型

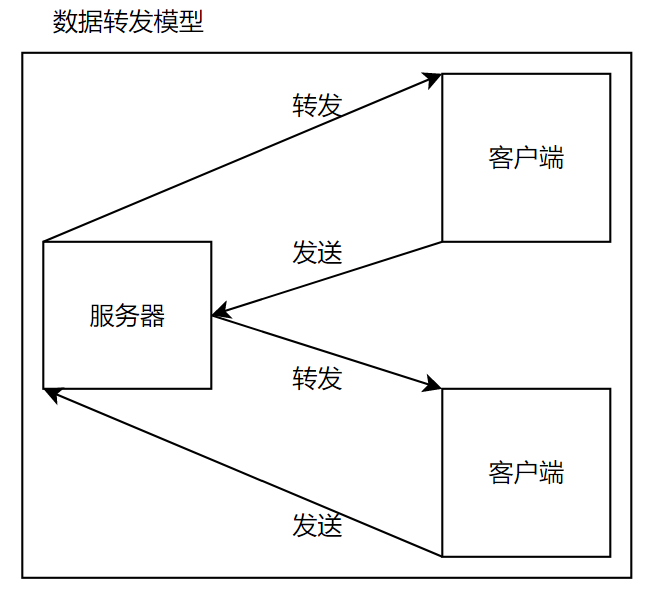


图 5.2 数据转发模型

从图 5.2 数据转发模型可以看出，数据转发模型用于多个客户端之间的数据转发。由于是S/C网络架构，客户端之间的数据必须通过服务器进行转发，便引用了该模型。数据转发模型在处理上，可以单方向转发，即客户端收到另一个客户端的消失时不会反馈，也可以在接收到信息之后发送确认反馈，反馈确认信息可以保证数据的到达状态。

数据转发模型被用于发送消息和文件传输功能上，其中发送消息采用了单向转发，即接收方不反馈。文件传输功能采用双向反馈，即接收到文件传输请求时，反馈接收状态，如是否接收文件、是否成功接收、是否接收完成等反馈确认信息。

## 交互协议结构

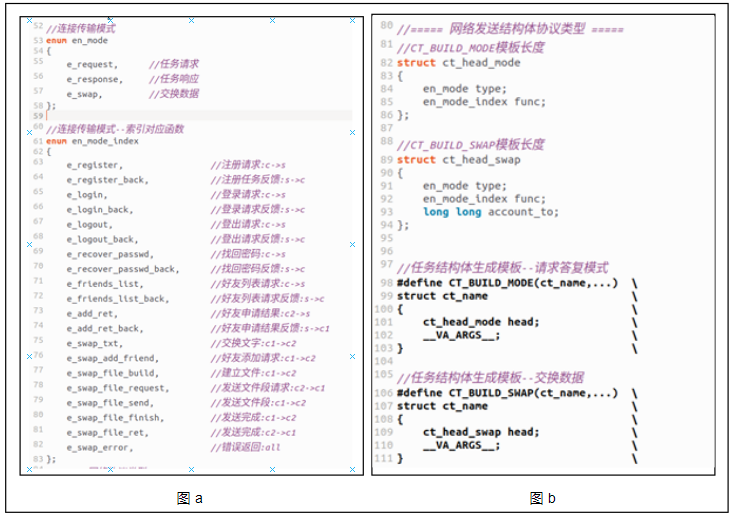


图 5.3 交互协议结构

在图 5.3 交互协议结构中，图a为协议类型与任务类型枚举，图b为交互协议传输结构体。传输结构体是网络编程中较为常用的数据打包手段，主要用于将信息放入结构体中，并将结构体转为二进制字节数组通过套接字发出，接收方接收到二进制字节数据后转成对应的结构体类型，从而完成数据打包。

从图b可以看出，结构体分为两类，其中ct\_head\_mode为请求响应模型的传输结构体，携带en\_mode与en\_mode\_index两个枚举类型的成员,分别用于接收方区分协议类型与请求任务类型。

ct\_head\_swap为数据转发模型的传输结构体，于请求响应模型结构类似，相比于请求响应模型结构体多了一个long long account\_to的成员变量，该成员用于表示数据转发的目标账号，当服务器接收到客户端的转发任务时，可快速锁定目标账号并转发。

# 实测结果与讨论

## 章节概要

本章将展示该程序的测试结果，测试的内容主要是客户端的各个UI界面，其中包括登录与注册窗口、错误信息反馈窗口、好友列表与聊天窗口、发送消失时聊天窗口的变化等客户端窗口界面内容。在数据交换测试中，会依次对比发送、接收两端对表情包与文件的不同处理。

## 登录窗口测试



图 6.1 登录测试

在图 6.1 登录测试中，图a为登录界面，图b为注册界面。

登录窗口与注册窗口都相对简单，其中登录窗口的输入密码部分使用了掩码进行防护，而设置密码与输入账号时采用显示模式，让用户可以清晰查看输入内容以防止错误。

在UI设计上，整个程序都采用了圆滑无边框设计，并增加暗黄色背景图，自定义的个性控件，其独特风格会贯穿整个程序，使UI保持统一风格。在UI控件的设计上，为了与背景融合，全部采用透明背景设计，与主流的UI界面设计不同。虽然背景透明，但各类控件依旧保证着显著特征，在不影响用户使用上增加个性化元素。



图 6.2 弹窗测试

在图 6.2 弹窗测试中，图a为注册申请反馈窗口，图b为登录失败弹窗。两者均为操作时的反馈窗口，用于提示用户对应操作反馈的结果，不管是操作成功还是操作失败都会得到反馈。

## 聊天窗口测试

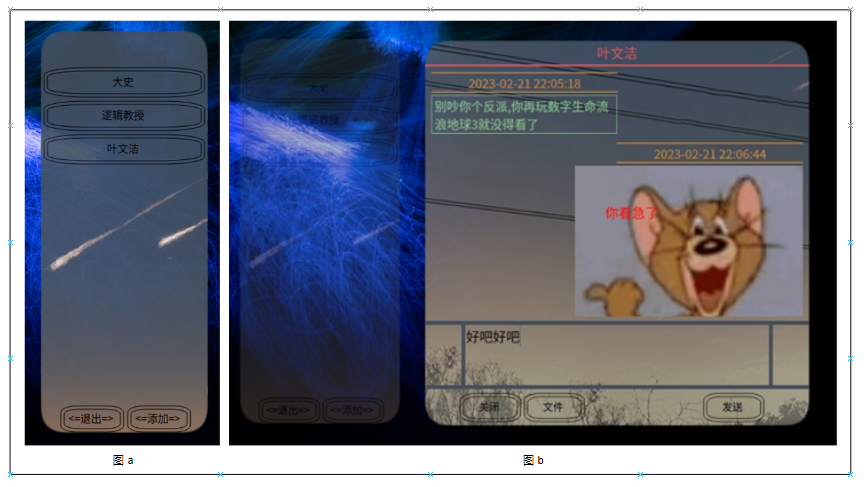


图 6.3 聊天窗口测试

在图 6.3 聊天窗口测试中，图a为好友列表窗口，图b为聊天窗口。其中好友列表窗口列出了所有好友，每位好友对应一个好友按钮以及聊天窗口，点击好友按钮会弹出聊天窗口。

从图b可以看出，聊天窗口分为显示区和输入区，红线与第一条蓝线为显示区，两根蓝线中间为输入区。在显示区可以显示接收到的消息以及表情包，发送区则可以输入文件并发送。当需要发送表情包时，只需要将表情包拖入到显示区即可发送。

从图b可以看出，当聊天窗口弹出时，好友列表几乎处于透明状态，其透明度的降低有利于用于将注意力转移到聊天窗口中。在整个程序中，只要窗口没有鼠标焦点，其透明度就会降低，此设计的初衷是帮助用户排除其他窗口的信息干扰，但需要是又可以快速调出。

## 文件进度测试



图 6.4文件进度测试

在图 6.4文件进度测试中，图a为接收方文件进度，图b为发送方文件进度。可以看出发送文件时，显示区窗口会右对齐进度条，并显示文件路径、发送当前进度，文件发送状态等信息。接收方除窗口对齐方式为左对齐之外，与发送方显示的信息格式几乎一致，其中显示的文件路径是保存路径。

## 好友申请测试



图 6.5 申请发送测试

在图 6.5 申请发送测试中，图a为聊天窗口中的申请好友按钮，图b为申请好友窗口。申请好友的方式为点击申请按钮，弹出申请窗口之后在添加输入框输入希望添加的好友账号，点击发送则向服务器发送申请好友的请求。

在图b中，第一个浅白色的输入框为本机的账号，当别人希望添加你为好友时，可通过输入该账号申请添加你为好友。



图 6.6 申请反馈测试

在图 6.6 申请反馈测试中，图a为好友申请成功的反馈，图b为申请成功之后好友列表的改动。当申请好友请求发出时，我们等待对方的确认，对方不管是同意或者是不同意都活反馈一个提示消息。本次测试为，当好友同意了请求，反馈了成功添加的窗口，当用户关闭窗口时，客户端重新向服务器请求好友列表信息，用于更新追加的好友按钮。

更新请求是双向的，不管是好友申请方还是发送方，在服务器接收到添加反馈时，同时向双方客户端下发好友列表信息，用于更新客户端的好友列表窗口。

# 总结和展望

即时通信软件是目前最火爆的软件开发趋势之一，即时通信包含的内容如聊天消息、语音通话、视频通话、录像直播等多种内容。除此之外，如今的聊天软件更是集成了各种功能，如好友动态、朋友圈、好友扩列、摇一摇聊天等众多社交方式，以及成为了现代人不可或缺的工具。

在功能之外，各类软件也都纷纷希望留住用户，收集个人信息，启用智能内容推荐增强用户依赖性，添加越来越多的功能，如在线支付功能、会员功能等，并引入小程序应用等扩展性非常强的功能，打造各自的生态坏境以留住用户。站稳市场之后，各类软件都开始对用户推送大量广告，在用户体验上大打折扣。

对于越来越多功能的聊天软件，越来越难体会到人与人之间纯粹的通信，总是被各类功能和广告所影响，我希望在未来开发一款纯粹聊天的软件，抛开各种以商业为主的功能，回归聊天的本质。

该软件是可以想象的，如保留了基本的通行功能，如消息、语音、视频通话通信等，提供个性化的标签以及朋友圈等熟悉的社交功能。在此之外，将打造一个拥有话题的聊天通道，该通道是对所有用户公开的公共频道。可以将话题分为各个领域，可以进入自己感兴趣的领域与拥有同意爱好的人共同探讨，如游戏、英语、数学、编程、通信、化学等各类话题频道。全公开频道对话才是理想聊天软件的趋势，如视频直播一般，只对话题感兴趣，而无论在与谁对话。

这一款理想的通信软件应该要足够的方便，可以通过各类渠道随时进入话题当中，如支持在手机、电脑、平板、智能手表等设备上使用，可以在不同的操作系统安装，并可在桌面程序、手机应用、浏览器等多种方式启动聊天功能。支持个人用户在内网部署通信软件，支持内网穿透技术解决大流量服务器部署问题，以离散的方式建立共同的话题。

此理想软件的实行，需要有足够的技术支持，以及找到拥有共同理想的同伴，在此之前仍需努力学习。

# 参考文献

1. Stanley B.LippmanBarbara E.MooJosée.C++primer 第三版[M]. 广东:中国电力出版社,2002:102-207.
2. 谢希仁.计算机网络[第7版] [M].北京:电子工业出版社,2017:303-359.
3. 鸟哥.鸟哥的Linux私房菜-基础篇 第四版[M].北京:[人民邮电出版社](https://book.douban.com/press/2609),2018:102-302.
4. 鸟哥.鸟哥的Linux私房菜服务器架设篇[M].北京:[人民邮电出版社](https://book.douban.com/press/2609),2008:56-97.
5. W.，Richard，Stevens .UNIX环境高级编程中文第三版[M].北京:[人民邮电出版社](https://book.douban.com/press/2609),2019:791-921.
6. W.，Richard，Stevens .UNIX网络编程卷1中文版[M].北京:[人民邮电出版社](https://book.douban.com/press/2609),2010:335-408.
7. 侯捷. STL源码剖析[M].湖北:华中科技大学出版社.2002,34-36.
8. Matthew H.Austern.泛型编程与STL[M].河北:中国电力出版社.2003:91-106.
9. 苏瑞、李牧.C语言程序设计[M].辽宁:大连理工大学出版社.2001:331-401.
10. 谭浩强.C语言程序设计(第五版)[M].北京:清华大学出版社.2017.
11. 马丁.代码整洁之道[M].北京:[人民邮电出版社](https://book.douban.com/press/2609).2011,229-302.
12. Martin Fowle.重构：改善既有代码的设计[M]. 北京:[人民邮电出版社](https://book.douban.com/press/2609).2011:77-92.
13. 徐胜昭.基于C\_S模式的局域网络聊天软件系统的设计与实现[D].电子科技大学,2012:1-9.
14. 董敏.基于Linux的网络聊天系统的设计[J].南京航空航天大学.2017:5-8.
15. 邓亚君.基于Python的网络聊天室设计[J].西华大学,2018:3-15.

# 致谢

四年的学习生活即将结束，回首往事，难以忘怀在这四年的学习和生活中给予我关怀和支持的老师和同学们。

我能够顺利地完成本科阶段的学习，首先要感谢所有任课老师、指导老师、热心相助的同学、康概解疑的前辈与相互帮衬的同事，感谢他们的乐于助人、无私奉献的精神。

在开始编写论文时遇到了很多的困难，而对于如何解决这些困难我却一无所知，辛亏得到了指导老师的悉心教导与专业课程的任课老师耐心付出才最终写下这一份论文。在一次次的遇到困难与解决困难的过程中我学到了很多，对于从无到有的知识累计与逐步获得成就感的学习过程感到非常高兴。

在撰写论文的时间里，正好也是实习期间，在实习时认识到了很多前辈与同事。在专业知识的实际运用时，常常遇到不懂的地方，当遇到疑惑时向前辈请教都得到了他们慷慨的解答与帮助。他们的帮助让我认识到实际运用与理论的差距，而这些知识与经验最终都帮助我完成了毕设的课题，也帮助我对人生这一重大课题做了进一步解答，在此非常感谢他们的热心帮助。

# 附录 A 完整的转发协议文件

#ifndef WEB\_PROTOCOL\_H

#define WEB\_PROTOCOL\_H

//#include "web\_protocol.h"

//===== 解析结构体 =====

//解体结构体信息

#include <string>

//结构体转string

template <class T\_ct>

static std::string to\_str(T\_ct ct)

{ return std::string((char\*)&ct,sizeof(T\_ct)); }

//string转结构体

template <class T\_ct>

static void to\_ct(const std::string &str,T\_ct &ct)

{ ct = \*(T\_ct\*)str.c\_str(); }

//===== 解析结构体 =====

//===== 网络协议类型 =====

//连接传输模式

enum en\_mode

{

e\_request, //任务请求

e\_response, //任务响应

e\_swap, //交换数据

};

//文件转发类型

enum en\_build\_file

{

e\_spic, //图片表情包

e\_file, //普通文件

};

//文件转发类型

enum en\_swap\_error

{

e\_error\_disconnect, //目标未连接:s->(c1,c2)

e\_error\_swap, //转发失败:s->(c1,c2)

e\_error\_build, //建立文件-错误反馈 :s->(c1)

e\_error\_send, //发送文件段-错误反馈 :s->(c1)

e\_error\_request, //发送文件段-错误反馈 :s->(c2)

e\_error\_finish, //发送完成-错误反馈 :s->(c1)

e\_error\_finish\_back, //接收文件完整性错误 :s->(c2)

};

//连接传输模式--索引对应函数

enum en\_mode\_index

{

e\_register, //注册请求:c->s

e\_register\_back, //注册任务反馈:s->c

e\_login, //登录请求:c->s

e\_login\_back, //登录请求反馈:s->c

e\_logout, //登出请求:c->s

e\_logout\_back, //登出请求反馈:s->c

e\_recover\_passwd, //找回密码:c->s

e\_recover\_passwd\_back, //找回密码反馈:s->c

e\_friends\_list, //好友列表请求:c->s

e\_friends\_list\_back, //好友列表请求反馈:s->c

e\_add\_ret, //好友申请结果:c2->s

e\_add\_ret\_back, //好友申请结果反馈:s->c1

e\_swap\_txt, //交换文字:c1->c2

e\_swap\_add\_friend, //好友添加请求:c1->c2

e\_swap\_file\_build, //建立文件:c1->c2

e\_swap\_file\_request, //发送文件段请求:c2->c1

e\_swap\_file\_send, //发送文件段:c1->c2

e\_swap\_file\_finish, //发送完成:c1->c2

e\_swap\_file\_ret, //发送完成:c2->c1

e\_swap\_error, //错误返回:all

};

//===== 网络协议类型 =====

//===== 网络发送结构体协议类型 =====

//CT\_BUILD\_MODE模板长度

struct ct\_head\_mode

{

en\_mode type;

en\_mode\_index func;

};

//CT\_BUILD\_SWAP模板长度

struct ct\_head\_swap

{

en\_mode type;

en\_mode\_index func;

long long account\_to;

};

//任务结构体生成模板--请求答复模式

#define CT\_BUILD\_MODE(ct\_name,...) \

struct ct\_name \

{ \

ct\_head\_mode head; \

\_\_VA\_ARGS\_\_; \

} \

//任务结构体生成模板--交换数据

#define CT\_BUILD\_SWAP(ct\_name,...) \

struct ct\_name \

{ \

ct\_head\_swap head; \

\_\_VA\_ARGS\_\_; \

} \

//快速声明结构体--请求(参数1：变量名，参数2：半截枚举)

#define MAKE\_CT\_REQ(in\_name,in\_type) \

ct\_##in\_type in\_name; \

in\_name.head.type = en\_mode::e\_request; \

in\_name.head.func = en\_mode\_index::e\_##in\_type \

//快速声明结构体--应答(参数1：变量名，参数2：半截枚举)

#define MAKE\_CT\_RES(in\_name,in\_type) \

ct\_##in\_type in\_name; \

in\_name.head.type = en\_mode::e\_response; \

in\_name.head.func = en\_mode\_index::e\_##in\_type \

//快速声明结构体--交换(参数1：变量名，参数2：半截枚举，参数3：目标账号)

#define MAKE\_CT\_SWAP(in\_name,in\_type,in\_account) \

ct\_##in\_type in\_name; \

in\_name.head.type = en\_mode::e\_swap; \

in\_name.head.func = en\_mode\_index::e\_##in\_type; \

in\_name.head.account\_to = in\_account \

//==客户与服务器==

//注册请求

CT\_BUILD\_MODE(ct\_register,

char passwd[64];

char name[64];

);

//注册任务反馈

CT\_BUILD\_MODE(ct\_register\_back,

long long account;

bool is\_success;

char passwd[64];

);

//登录请求

CT\_BUILD\_MODE(ct\_login,

long long account;

char passwd[64];

);

//登录请求反馈

CT\_BUILD\_MODE(ct\_login\_back,

long long account;

bool is\_success;

);

//登出请求

CT\_BUILD\_MODE(ct\_logout,

long long account;

);

//登出请求反馈

CT\_BUILD\_MODE(ct\_logout\_back,

long long account;

bool is\_success;

);

//找回密码

CT\_BUILD\_MODE(ct\_recover\_passwd,

long long account;

);

//找回密码反馈

CT\_BUILD\_MODE(ct\_recover\_passwd\_back,

long long account;

bool is\_success;

char passwd[64];

);

//好友列表请求

CT\_BUILD\_MODE(ct\_friends\_list,

long long account;

);

//好友列表请求反馈

CT\_BUILD\_MODE(ct\_friends\_list\_back,

bool is\_end;

long long account;

char name[64];

);

//好友添加请求结果

CT\_BUILD\_MODE(ct\_add\_ret,

long long account\_from;

long long account\_to;

bool is\_agree;

);

//好友添加请求结果

CT\_BUILD\_MODE(ct\_add\_ret\_back,

long long account\_from;

bool is\_agree;

bool is\_self;

);

//==客户与服务器==

//==客户与客户==

//交换文字

CT\_BUILD\_SWAP(ct\_swap\_txt,

long long account\_from;

char buf\_txt[1024];

);

//好友添加请求

CT\_BUILD\_SWAP(ct\_swap\_add\_friend,

long long account\_from;

);

//建立文件

CT\_BUILD\_SWAP(ct\_swap\_file\_build,

long long account\_from;

long long size\_block; //发送块大小

long long size\_file; //文件总大小

en\_build\_file type;

char path[255];

char filename[255];

);

//发送文件段请求

CT\_BUILD\_SWAP(ct\_swap\_file\_request,

long long account\_from;

long long block\_count;

char filename[255];

);

//发送文件段

CT\_BUILD\_SWAP(ct\_swap\_file\_send,

long long account\_from;

long long off\_file; //当前文件偏移

long long size\_buf; //本次发送的真实字节

bool is\_next;

char filename[255];

char buf[4096];

);

//发送完成

CT\_BUILD\_SWAP(ct\_swap\_file\_finish,

long long account\_from;

char filename[255];

);

//发送完成反馈

CT\_BUILD\_SWAP(ct\_swap\_file\_ret,

long long account\_from;

en\_build\_file type;

bool is\_success;

char filename[255];

);

//错误反馈

CT\_BUILD\_SWAP(ct\_swap\_error,

long long account\_from;

en\_swap\_error err;

en\_mode\_index swap\_func;

);

//==客户与客户==

#endif // WEB\_PROTOCOL\_H

了解一份完整的通信协议非常重要，代码中利用CT\_BUILD\_MODE和CT\_BUILD\_SWAP宏代码快速生成请求响应模型和数据转发模型的结构体代码。

# 附录 B 服务器任务分发代码

1. 头文件任务声明

//===== 任务函数 =====

map<en\_mode\_index,func\_task> map\_task\_func; //任务函数查询容器

void task\_register(const web\_sock& sock, const string& meg); //账号注册

void task\_login(const web\_sock& sock, const string& meg); //登录请求

void task\_logout(const web\_sock& sock, const string& meg); //登出请求

void task\_recover\_passwd(const web\_sock& sock, const string& meg); //忘记密码

void task\_friends\_list(const web\_sock& sock, const string& meg); //好友列表

void task\_swap(const web\_sock& sock, const string& meg); //数据交换

void task\_add\_ret(const web\_sock& sock, const string& meg); //好友申请

1. 构造函数添加任务到容器

//===== 添加任务函数到容器 =====

//==原型:

//map\_task\_func.insert(pair<en\_mode\_index,func\_task>

// (en\_mode\_index::e\_login,bind(&ux\_web\_server::task\_login,this,\_1,\_2)));

//===== 添加任务函数到容器 =====

#define MAP\_TASK\_ADD(func) \

map\_task\_func.insert(pair<en\_mode\_index,func\_task> \

(e\_##func,bind(&ux\_web\_server::task\_##func,this,\_1,\_2)))

MAP\_TASK\_ADD(register); //账号注册

MAP\_TASK\_ADD(login); //登录请求名

MAP\_TASK\_ADD(logout); //登出请求

MAP\_TASK\_ADD(recover\_passwd); //忘记密码

MAP\_TASK\_ADD(friends\_list); //好友列表

MAP\_TASK\_ADD(add\_ret); //好友申请

//===== 添加任务函数到容器 =====

1. 事件分发函数

void ux\_web\_server::on\_message(const web\_sock &sock, const string &meg)

{

//执行匹配的任务函数

ct\_head\_mode ct;

to\_ct(meg,ct);

if(ct.type == en\_mode::e\_swap) //进入交换函数

{ task\_swap(sock,meg); }

else if(ct.type == en\_mode::e\_request) //进入应答函数

{

auto it\_func = map\_task\_func.find(ct.func);

if(it\_func != map\_task\_func.end())

{ (std::bind(it\_func->second,sock,meg))(); }

else { vloge("not find func: e\_request func"); }

}

else //未发现协议

{ vloge("on\_message not find: " vv(ct.type) vv(ct.func)); }

}

服务器任务分发代码分为三个部分，分别是第一部分，头文件任务声明，该部分在服务器头文件列出所有任务的处理函数。

第二分部，构造函数添加任务到容器，这段代码在服务器构造函数执行时，插入到map容器中，其中任务处理函数作为Value，任务类型为Key。

第三部分，事件分发函数，每当套接字接收到消息时，分解出任务类型，并将任务类型通过map查询与之绑定的任务处理函数，如果存在则执行任务处理函数完成事件分发。

# 附录 C 客户端的任务转发代码

//发送--用户输入信息

connect(v\_friends\_list,&wid\_friends\_list::fa\_send\_news,this,

[=](en\_info en,long long account,QString info){

if(en == en\_info::e\_send\_txt)

{

vlogf("e\_send\_txt" vv(account) vv(qtos(info)));

v\_net->ask\_swap\_txt(account,info);

}

else if(en == en\_info::e\_send\_pic)

{

vlogf("e\_send\_pic" vv(account) vv(qtos(info)));

QFileInfo info\_file(info);

v\_net->ask\_swap\_file(account,info\_file.fileName(),info,en\_build\_file::e\_spic);

}

else if(en == en\_info::e\_send\_file)

{

vlogf("e\_send\_file" vv(account) vv(qtos(info)));

QFileInfo info\_file(info);

v\_net->ask\_swap\_file(account,info\_file.fileName(),info,en\_build\_file::e\_file);

}

});

//发送--添加好友请求

connect(v\_friends\_list,&wid\_friends\_list::fa\_add\_friend,this,

[=](long long account){

v\_net->ask\_swap\_add\_friend(account);

});

//发送--登陆

connect(v\_login,&wid\_login::fa\_login,this,

[=](long long account,QString passwd){

vlogf("fa\_login" vv(account) vv(qtos(passwd)));

v\_net->ask\_login(account,passwd);

});

//发送--注册

connect(v\_login,&wid\_login::fa\_register,this,

[=](QString name,QString passwd){

vlogf("fa\_register" vv(qtos(passwd)) vv(qtos(name)));

v\_net->ask\_register(passwd,name);

});

//===== 网络反馈 =====

//==服务器响应

connect(v\_net,&net\_connect::fa\_open,this,[=](){

vlogf("fa\_open");//

});

connect(v\_net,&net\_connect::fa\_close,this,[=](){

vlogf("fa\_close");//

wid\_dialog dia("网络连接断开[确认并重新连接]");

dia.exec();

if(dia.get\_status())

{ v\_net->open\_connect(ip,port); }

else { exit(-1); }

});

connect(v\_net,&net\_connect::fa\_register\_back,this,

[=](long long account,QString passwd,bool ok){

vlogf("fa\_register" vv(account) vv(qtos(passwd)) vv(ok));//

v\_login->show\_register\_back(ok,account);

});

connect(v\_net,&net\_connect::fa\_login\_back,this,

[=](long long account,bool ok){

vlogf("fa\_login\_back" vv(account) vv(ok));

if(ok) { v\_net->ask\_friends\_list(v\_net->get\_account()); }

else

{

wid\_dialog dia("登录失败[帐号或者密码错误]");

dia.exec();

}

});

connect(v\_net,&net\_connect::fa\_logout\_back,this,

[=](long long account,bool ok){

vlogf("fa\_logout\_back" vv(account) vv(ok));

if(ok) v\_net->close\_connect();

});

connect(v\_net,&net\_connect::fa\_recover\_passwd\_back,this,

[=](long long account,QString passwd,bool ok){

vlogf("fa\_recover\_passwd\_back" vv(account) vv(qtos(passwd)) vv(ok));//

if(ok) v\_net->close\_connect();

});

connect(v\_net,&net\_connect::fa\_friends\_list\_back,this,

[=](QMap<long long,string> map){

//第一次进入时的窗口状态切换

if(v\_login->isHidden() == false)

{

v\_friends\_list->set\_account(v\_net->get\_account());

v\_friends\_list->move(v\_login->pos());

v\_login->close();

}

v\_friends\_list->into\_friends(map);

v\_friends\_list->show();

});

connect(v\_net,&net\_connect::fa\_add\_ret\_back,this,

[=](long long account,bool ok,bool self){

if(self == true) { v\_net->ask\_friends\_list(v\_net->get\_account()); }

else

{

QString info;

if(ok) { info = "成功添加账号: " + QString::number(account); }

else { info = "对方不同意添加为好友: " + QString::number(account); }

wid\_dialog temp(info);

temp.exec();

if(ok) { v\_net->ask\_friends\_list(v\_net->get\_account()); }

}

});

//交换函数

connect(v\_net,&net\_connect::fa\_swap\_txt,this,

[=](long long account\_from,QString txt){

vlogf("fa\_prog\_send" vv(account\_from) vv(qtos(txt)));//

v\_friends\_list->into\_news(en\_info::e\_send\_txt,account\_from,txt);

});

connect(v\_net,&net\_connect::fa\_swap\_add\_friend,this,

[=](long long account\_from){

QString info = "同意添加[ "+ QString::number(account\_from) +" ]为好友吗";

wid\_dialog temp(info);

temp.exec();

v\_net->ask\_add\_ret(account\_from,temp.get\_status());

});

connect(v\_net,&net\_connect::fa\_swap\_build,this,

[=](long long account\_from,QString filename,en\_build\_file type){

if(type == en\_build\_file::e\_file)

{

vlogf("fa\_prog\_send: " vv(account\_from) vv(qtos(filename)));//

v\_friends\_list->into\_news(en\_info::e\_send\_file,account\_from,filename);

}

});

connect(v\_net,&net\_connect::fa\_swap\_file\_finish,this,

[=](long long account\_from,QString filename,en\_build\_file type,bool is\_ok){

if(type == en\_build\_file::e\_file)

{

//文件进度(需要三个字段:文件名,进度值,是否完成(prog/finish))(分割符:##)

QString info = filename+"##100##1";

v\_friends\_list->into\_news(en\_info::e\_send\_file\_prog,account\_from,info);

}

else if(type == en\_build\_file::e\_spic)

{ v\_friends\_list->into\_news(en\_info::e\_send\_pic,account\_from,filename); }

vlogd("fa\_swap\_file\_finish" vv(account\_from) vv(qtos(filename)) vv(type) vv(is\_ok));//

});

connect(v\_net,&net\_connect::fa\_swap\_file\_ret,this,

[=](long long account\_from,QString filename,en\_build\_file type,bool is\_ok){

if(type == en\_build\_file::e\_file)

{

//文件进度(需要三个字段:文件名,进度值,是否完成(prog/finish))(分割符:##)

QString info = filename+"##100##1";

v\_friends\_list->into\_news(en\_info::e\_send\_file\_prog,account\_from,info);

}

vlogd("fa\_swap\_file\_ret" vv(account\_from) vv(qtos(filename)) vv(type) vv(is\_ok));//

});

connect(v\_net,&net\_connect::fa\_swap\_error,this,

[=](long long account\_from,en\_swap\_error err){

vlogf("fa\_swap\_error" vv(account\_from) vv(err));//

});

connect(v\_net,&net\_connect::fa\_prog\_send,this,

[=](QString filename,long long account\_from,int now){

//文件进度(需要三个字段:文件名,进度值,是否完成(prog/finish))(分割符:##)

QString info = filename+"##"+QString::number(now)+"##0";

v\_friends\_list->into\_news(en\_info::e\_send\_file\_prog,account\_from,info);

});

connect(v\_net,&net\_connect::fa\_prog\_recv,this,

[=](QString filename,long long account\_from,int now){

//文件进度(需要三个字段:文件名,进度值,是否完成(prog/finish))(分割符:##)

QString info = filename+"##"+QString::number(now)+"##0";

v\_friends\_list->into\_news(en\_info::e\_send\_file\_prog,account\_from,info);

});

客户端的任务代码转发部分主要是由Qt的UI界面线程到网络连接线程的数据转发与任务封装。

在GUI程序中，由于Widget类组件事件循环中等待，网络连接线程直接操控Widget类组件会使得程序直接崩溃，必须通过Qt的connect信号转发，或者自定义事件触发事件循环才可以将网络线程的数据发送到GUI线程。客户端任务转发部分代码就是Qt主线程跟网络连接线程的胶水层，非常重要。

广西民族师范学院本科毕业论文(设计)诚信保证书

（学生用）

对于毕业论文（设计）《基于QT的广域网通讯软件开发》，本人郑重承诺：其内容真实、可靠，系本人在 郭敏杰 教师的指导下独立完成；如果被查出有弄虚作假、抄袭剽窃行为，本人愿承担全部责任。

承诺者（签名）：

年 月 日