基于中央定位服务器的 P2P 网络聊天系统

设计报告

2017011527 自 76 吴紫屹

2019 年 12 月 14 日

一、总体介绍

1.1 项目需求分析

本次计网大作业需要实现的是基于中央服务器的聊天程序，其中用户和中央服务器的连接为 C/S 架构，用于查询其他用户是否在线和其 IP 地址；而用户之间的连接为 P2P 架构，用于聊天通信。

1.2 实现功能概览

我使用 Python 语言完成了本次大作业的所有程序编写，界面则是基于 PyQt5 进行的设计，且本程序未使用 UDP 相关连接。大作业完成的功能列举如下：

- **必做内容：**
  - 账号登录上下线
  - 维护通讯录，查询好友是否在线
  - P2P 文字通信（基于 TCP）
  - 大文件/图片传输（10MB 以上）
  - 友好的 UI 界面
- **选做内容：**
  - 语音和视频通话（基于 TCP，动态调整分辨率）
  - P2P 群聊（仅实现文字，未实现 P2P 文件分发）
- **其他功能：**
  - 发送 emoji 表情
  - 语音输入，直接发送
  - 语音输入，转文字发送
  - 模仿中央服务器编写自己的服务器，支持单机同时运行多个聊天客户端与之连接，相互通信，方便调试

本报告之后章节的安排如下：第二章将主要介绍程序的底层实现，例如使用到的类的定义；第三章将主要介绍程序的通讯协议；第四章将详述各子功能的具体实现方法；最后则是结果和分析。有关 UI 使用方法的介绍，我会在 README 文件中进行详细地阐述。

二、底层实现

本章节将主要介绍几个关键类的定义和实现。值得一提的是，部分类的成员变量仅仅是为了完整性才定义的，但是由于时间有限所以并没有用到。

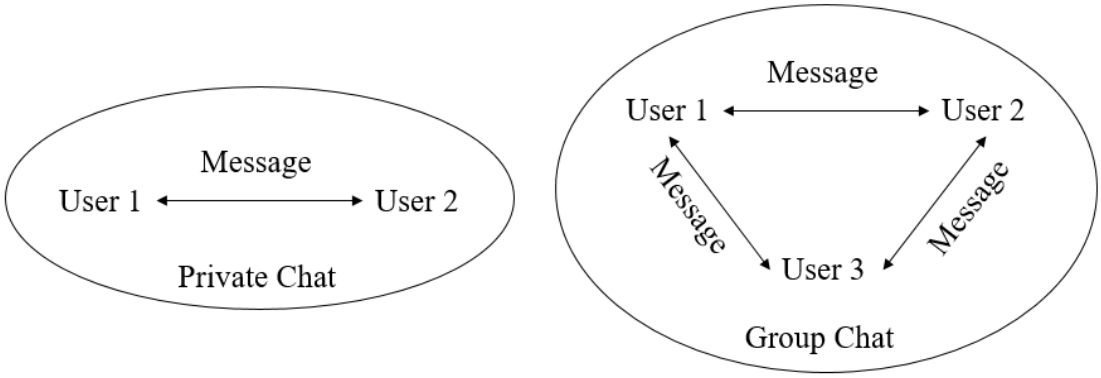

图 1 程序关键类及相关关系

2.1 用户

User 类是每一个聊天系统使用者在程序中的体现，包含以下成员变量：

- 学号：作为用户的唯一标识。这里有一个很重要的假设就是每一个用户的学号是不会重复的，整个程序的许多判断都是建构在这个假设上的
- 昵称：实际使用中直接用学号指代
- IP 地址
- 端口号
- 头像：实际使用中所有用户都使用默认头像

2.2 消息

Message 类是所有用户间通讯的信息载体，细分为文字消息、图片消息、文件消息三种：

- 共有成员变量：
  - 发送者学号
  - 接收者学号
  - 发送时间
  - 消息类型
  - 发送者头像：用于聊天框中显示
- 文字消息：
  - 内容字符串
- 图片/文件消息共有：
  - 路径
  - 大小

2.3 聊天

Chat 类将该聊天涉及到的用户、socket 等封装起来，并且实现了各类消息的发送接收和历史消息储存，分为私聊、群聊、视频聊天三种：

- 共有成员变量：
  - 本用户
  - 其余用户（在群聊中为以学号为键用户为值的 Python dict 数据类型，其余为单个用户）
  - 名称：作为聊天的唯一标识，命名方法为将涉及到的所有用户（包括自己）的学号以升序排序后用‘-’从中连接
  - 历史信息：用一个 Python list 数据类型简单实现
  - 由于程序中的每一个 Chat 都是作为子线程运行的，因此有用于标识聊天是否仍在进行的 flag 变量，以及与主界面线程进行通讯的信号槽变量
- 私聊：
  - socket：一个与对方持续连接的套接字
- 群聊：
  - sockets：以学号为键套接字为值的 Python dict 数据类型，与群聊中的每一个用户都维护一个持续连接的 socket
- 视频聊天：
  - 由于该功能是最后加的，因此为了不改变之前代码的架构，单独实现
  - 核心成员变量为视频和语音的发送及接收器，之后章节中会详细介绍

2.4 程序的功能实现思路

基于以上介绍的底层类我们可以简要阐述本聊天程序的实现思路。用户首先进行登录，登录成功则程序将实例化一个 User 类来存储个人信息；同时开一个子线程让一个 socket 监听任何其他用户发来的聊天请求，监听到则实例化一个 Chat 类，并使其作为一个子线程运行收发消息的功能；另一方面用户也可以主动查询其余用户并发起群聊，同样是作为一个子线程进行消息的收发；从 socket 中接收到的消息或想发送的消息则根据消息类型实例化一个 Message 类，并依通讯协议进行处理。

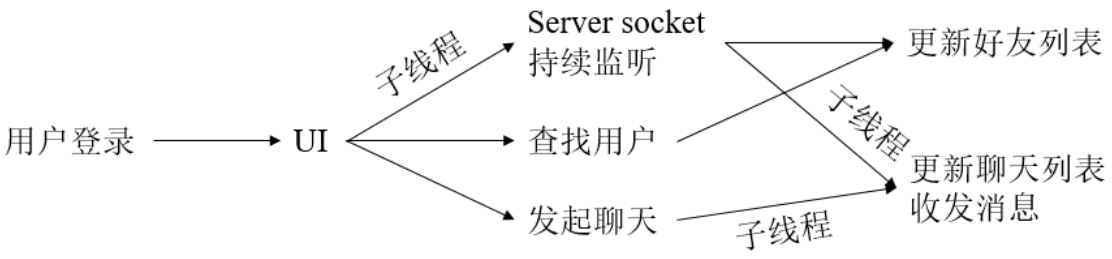

图 2 程序基本流程

三、通讯协议

本章节将主要介绍程序所使用的通讯协议。虽然 TCP 协议保证了可靠的数据传输，但是由于发送消息类型的多样化，针对不同的消息需要有不同的处理方式，这也就要求我们自行设计通信数据报格式。

3.1 数据包装

由于 TCP/IP 是流式协议，因此调用一次 socket.recv() 得到的结果并不一定是一次 socket.send() 所发送的内容。为了区分每一个完整的数据包，使得程序每次都可以正确地处理一个数据包，我借鉴[1]的做法对每一段发送的数据都在头部附上了数据长度的信息；接收时则先确认该数据的长度，然后只从缓冲区中读取此长度的数据，这样就保证了每次都能收到一个完整的数据包进行处理。

3.2 前导数据报

为了应对多种不同的消息类型，每次发送消息前我都会先发一个前导数据报来通知对方做出相应的接收准备。此外，创建聊天、好友登出等信息也是通过前导数据报通知的，只不过在这些情况下接收方不会准备接收后续数据。前导数据报的格式如下：

聊天名称_发送方监听端口_类型_发送方学号_内容

- 聊天名称：用于创建聊天或确认消息正确发送
- 发送方监听端口：由于中央服务器不提供用户端口，为方便单机调试故在此进行通知。实际使用中所有程序都用单一端口则该位无用
- 类型：标记该前导数据报的作用，例如“接下来要发送一个文件”或“对方用户发来视频聊天请求”等
- 发送方学号：用于确认消息正确发送
- 内容：取决于该数据报的类型，例如创建聊天请求中该位无用，而文件发送通知中该位包含文件名和文件大小

3.3 前导数据报种类及应对措施列举

0. 文字消息发送通知：检验发送方学号是否正确，提取字符串文本，更新聊天历史消息，消息框中显示

1. 图片消息发送通知：检验发送方学号是否正确，提取图片名称、大小，开始接收图片，更新聊天历史消息，消息框中显示

2. 文件消息发送通知：检验发送方学号是否正确，提取文件名称、大小，开始接收文件，更新聊天历史消息，消息框中显示

3. 私聊请求：检验发送方是否合理（例如学号是否格式正确），添加新的好友，

添加新的私聊，发送私聊确认回复
4. 群聊请求：检验发送方是否合理，添加新的群聊（与微信类似，群友并不会自动加为好友），发送群聊确认回复
5. 私聊确认回复：之前添加的私聊作为子线程开始运行
6. 群聊确认回复：之前添加的群聊作为子线程开始运行
7. 好友登出通知：将该好友相关信息、私聊删除（本程序认为 P2P 聊天若一方登出则直接删好友，不保存历史信息，毕竟数据库并不是本次大作业重点）
8. 好友删除通知：将该好友相关信息、私聊删除
9. 视频聊天请求：检验发送方学号是否正确，添加新的视频聊天，发送视频聊天确认回复
10. 视频聊天确认回复：之前添加的视频聊天作为子线程开始运行

四、详细设计

本章节将主要介绍各子功能的具体实现方法。涉及到 UI 操作方式的内容将在 README 中予以详述。

4.1 必做内容

4.1.1 账号登录上下线

当用户运行程序时首先弹出登录窗口，需输入用户名及密码（net2019）后点击登录。这时程序将会创建一个 socket 与中央服务器进行连接（此 socket 在程序运行期间一直保留，以下简称 central_socket），然后按照大作业 PPT 中所给指令格式进行发送、接收回复，进而确定用户是否登录成功。

登出操作类似，UI 界面右上角有一个登出按键，点击后 central_socket 将按规定格式发送命令，通知自己已登出。

4.1.2 维护通讯录，查询好友是否在线

用户输入学号并点击加好友，则 central_socket 将按照规定格式发送查询命令，并根据回复确定对方是否在线，若在线则加为好友、发起私聊请求。

为确认好友状态，程序将每隔一段时间使用 central_socket 查询所有好友是否下线、更新 IP 或标明已离线。

此外，用户还可以删除好友，若确认删除则与该好友相连的 socket 将发送好友删除通知，通知双方抹除对方所有痕迹。

4.1.3 P2P 文字通信（基于 TCP）

在二三章的基础上这一功能容易实现，假设用户 A 要向用户 B 发送一段文字，若 A、B 不是好友，则需要先通过 4.1.2 所述方式查询并加为好友、发起私聊，之后流程一样，A 发送 3.2 所述“文字消息发送通知”前导数据报并将文字字符串置于“内容”位，按 3.1 所述包装上数据长度后通过 socket 发送。B 接收到

前导数据报后按 3.3 所述方式进行接收、显示、保存。

4.1.4 大文件/图片传输

大文件传输与文字消息不同在于数据量大，无法通过一次 socket.send() 发送完成（对于那些可以一次发送的小文件同样按照这一方式发送，因为函数复用较为简洁）。假设用户 A 要向用户 B 发送一个文件，则 A 首先发送“文件消息发送通知”前导数据报并将文件名、文件大小置于“内容”位，B 收到通知后，在默认路径下创建一个同名空文件，准备接收。

此时 A 开始发送文件，每次读取该文件的 1024 个字节包装后发送，直至文件发送完毕。B 则每次接收并写入本地文件，直至已接收字节长度等于文件大小。最后还是照例显示在聊天框及更新历史消息记录。

图片的发送过程与文件类似，只不过在接收完毕后会把图片显示在聊天框而不仅仅是一个文件名保存通知。

4.1.5 友好的 UI 界面

详见 README 文档。

4.2 选做内容

4.2.1 语音和视频通话（基于 TCP，动态调整分辨率）

这部分的设计思路主要参考[2]，我首先编写了两个类实现调用 PC 摄像头及麦克风的功能，然后将它们分别封装进视频发送器和语音发送器两个类中，并编写了视频接收器和语音接收器两个类用于接收。则当用户 A 和用户 B 进行视频通话时，A 首先发送“视频聊天请求”，B 接收后若同意则回复“视频聊天确认”，双方皆实例化之前提到的视频、语音发送、接收器进行连接，开始通话。

具体而言，视频是通过快速发送连续的图片实现的。我在一个 while 循环中不断调用摄像头读取图片，压缩后发送。动态调整分辨率的实现是通过维护一个变量记录调用摄像头的频率，与一个设定好的理想帧率进行比较即可据此动态调整图片的分辨率大小。例如调用速率慢于理想帧率的 0.9 倍，则之后读取图片会 resize 到原图的 0.9 倍尺寸进行发送，以期达到理想帧率。每张图片的发送与 4.1.4 所述类似。接收方每接到一张图片就显示在屏幕上，只要帧率足够即可形成视频的效果。

音频方面，我发现音频的传输较快速，因此没有做动态调整，具体实现为边读取音频流边发送，接收方则将接收到的音频流写入麦克风中，就可以听到对方的声音。

4.2.2 P2P 群聊

群聊的实现思路很简单，就是每一位群友之间都维护一个 socket 连接，任一人发送消息就通过 socket 给所有人发一遍，发送方法完全按照私聊的方式。

较为麻烦的是创建群聊的过程，群主拥有所有群友的 IP 和学号因此可以建立 socket 连接，但是群友不一定互相认识，因此无法建立 socket 互联，也就无法互发消息。为了解决这一问题，我将每一位的群友的学号按 2.3 所述嵌入到群聊名称中，这样当一位用户接收到“群聊请求”时，就可以通过解析“群聊名称”得到所有群友的学号，继而向中央服务器查询 IP 建立 socket 连接。更具体而言，为了保证每一位群友之间不重复连接，用户只会向学号小于自己的群友发起 socket 连接请求，这就确保了每两人之间只会有一个 socket 连接。

虽然没实现 P2P 文件传输（课上所讲 Torrent 原理），不过通过一一发送的方式还是可以传输文件的，只不过速度比较感人。此外群聊不具有视频通话功能。

4.3 其他功能

4.3.1 发送 emoji 表情

实现极其简单，只需注意到 emoji 实际上可以按照 Unicode 编码，则发送 emoji 实际上和发送文字消息同理。

4.3.2 发送语音

如 4.2.1 所述我实现了一个麦克风调用采集音频的模块，因此发送语音只需要将采集到的音频写入文件，然后按照发送文件的方式发送即可。

4.3.3 语音转文字输入

通过调用百度语音识别 API[3]，对采集到的用户语音进行识别然后转换为文字，即可实现方便的语音输入功能。

4.3.4 自己的中央服务器

为了方便单机多程序的调试，我模仿助教提供的中央服务器自己编写了一款中央服务器，主要添加了端口号提供功能，无需修改代码中的硬编码即可同时运行多个聊天程序。

通信类型	客户端发送指令	服务器返还指令
登录	学号_密码_端口号 例：“2017011527_net2019_2333”	“lol”
查询好友状态	q 学号 例：“q2017011527”	IP 地址_端口号（在线）/“n”（不在线） 例：“192.168.1.1_2333”
下线	logout 学号 例：“logout2017011527”	“loo”

表 1 自制中央服务器指令表

服务器保持常开，并监听一个周知端口。每次收到连接请求就单开一个线程处理，并更新用户 IP、端口、在线情况。

五、结果及分析

本章节将主要展示部分功能的运行结果，某些不方便以图文形式展示的功能（语音输入）只能麻烦助教自行体验了。

5.1 必做内容

5.1.1 登录/登出

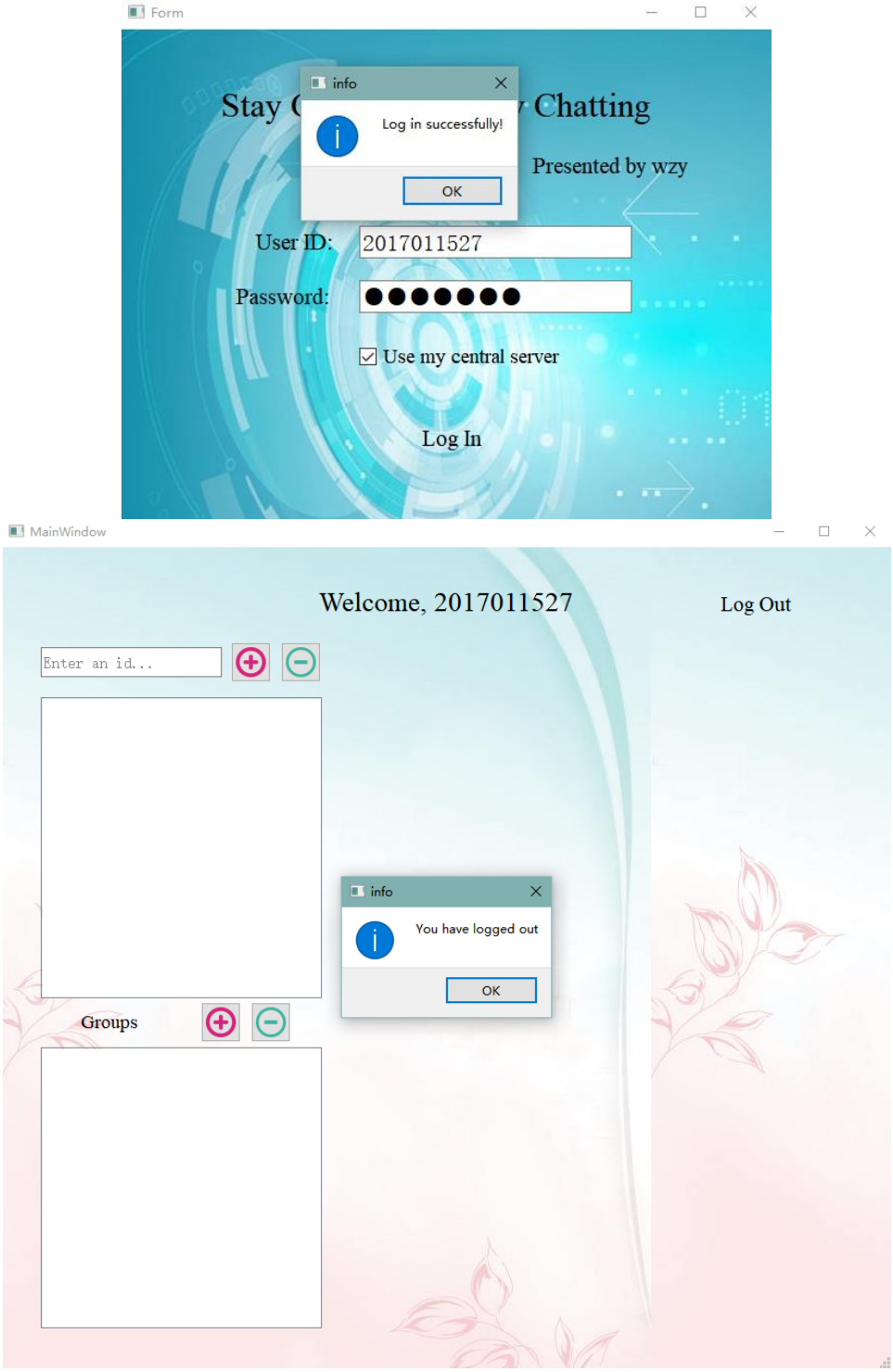

图 3 登录/登出

5.1.2 维护通讯录，查询好友状态

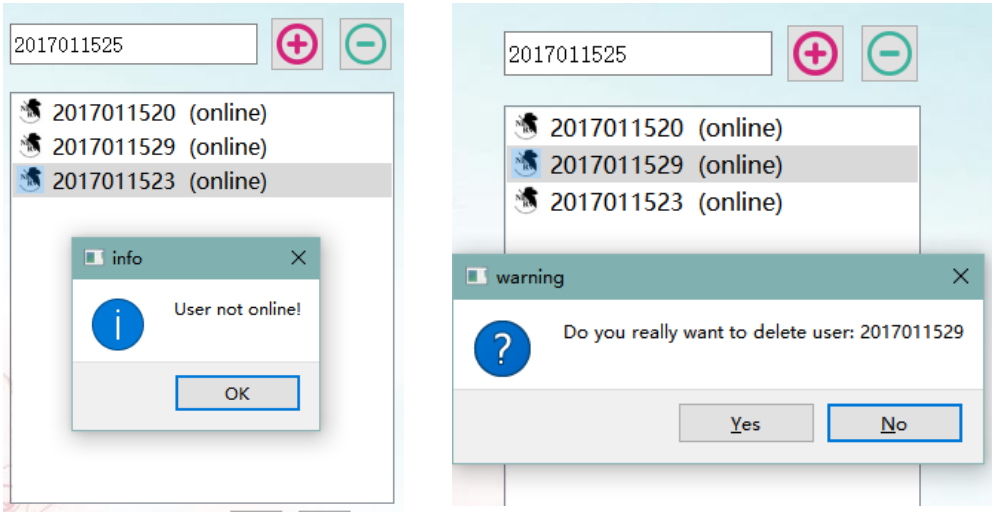

图 4 添加/删除好友

5.1.3 P2P 文字通信

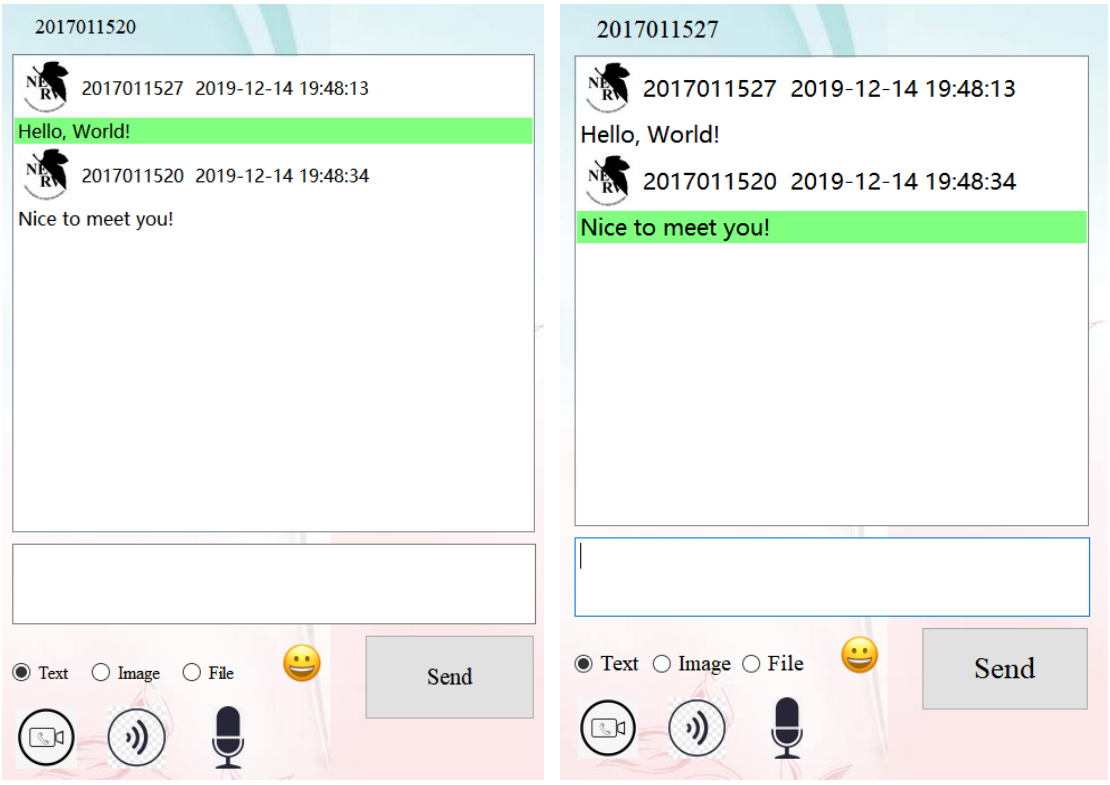

图 5 用户间文字聊天

5.1.4 大文件传输

以一个 20.9MB 的文件为例：

图 6 大文件传输

5.1.5 友善 UI 界面

详见 README 文档。

5.2 选做内容

5.2.1 语音和视频通话

在本机和舍友 PC 上进行通话，视频帧率稳定在 20fps+，单帧分辨率稳定在 720P：

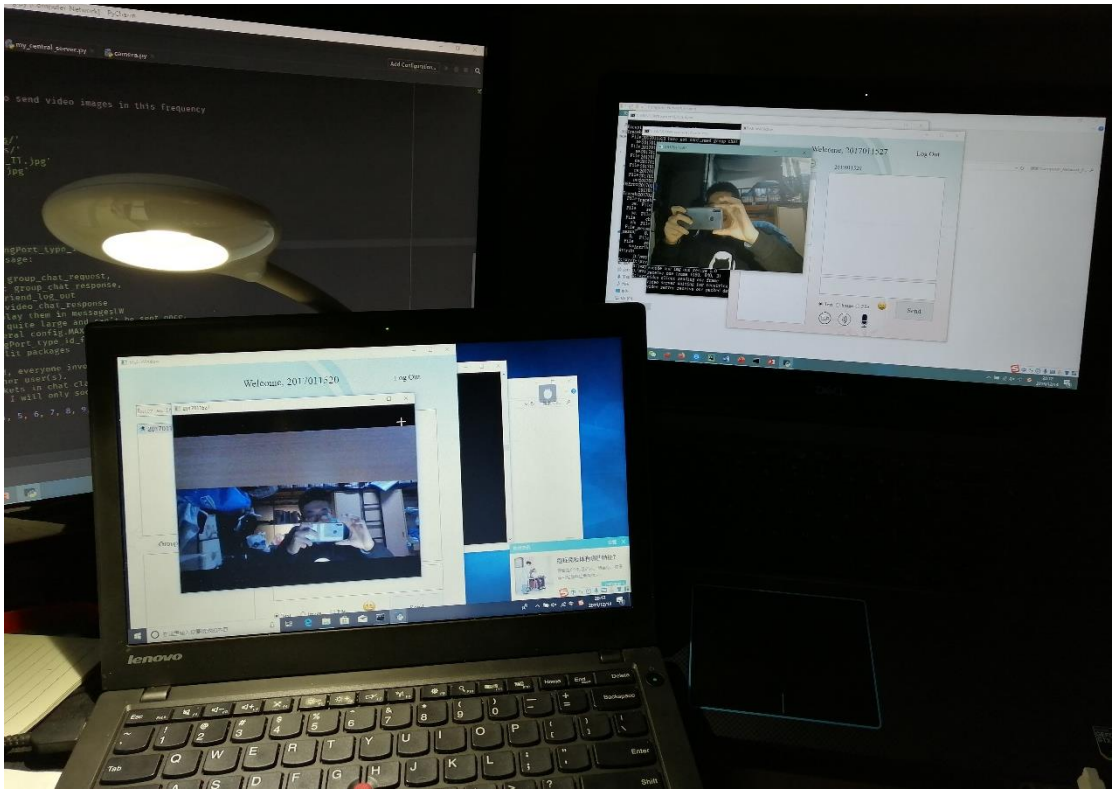

图 7 视频通话（精分现场.jpg）

5.2.2 P2P 群聊

以一个简单的三人群聊为例进行演示：

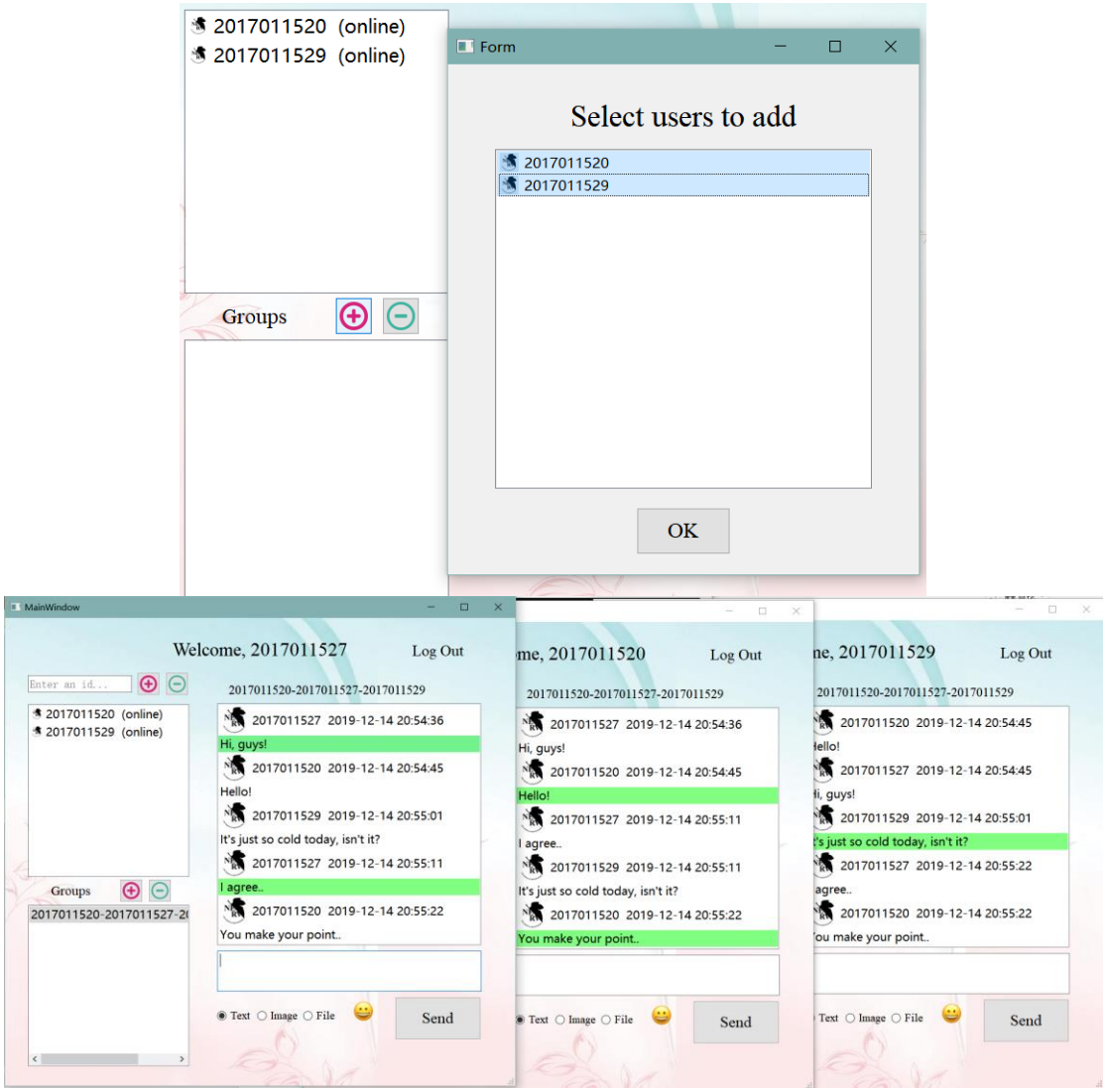

图 8 三人群聊

5.3 其他功能

5.3.1 发送 emoji

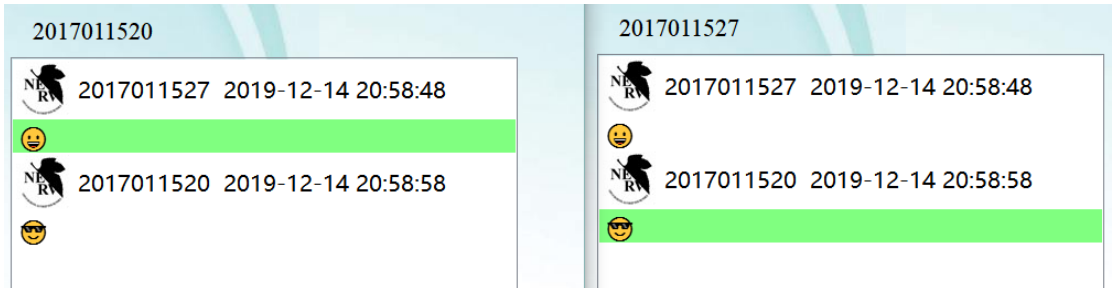

图 9 发送表情

5.3.2 发送语音

烦请助教自行体验。

5.3.3 语音输入

烦请助教自行体验。

5.3.4 自制中央服务器

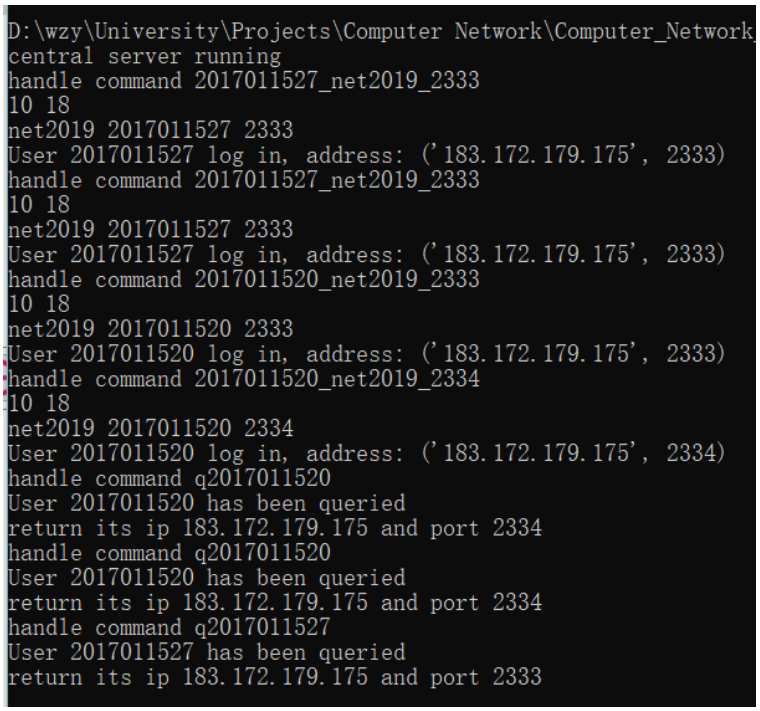

图 10 运行截图

可以看到，有两名用户登录，并查询好友

六、总结

本次大作业，是个人第一次接触网络编程相关的完整项目，掌握了许多 socket 通信编程的技巧，收获还是很大的，对于课上所学知识也进行了一次复习。不过，本次大作业主要还是聚焦于应用层，其余各层的运用可能还需将来进一步尝试。

总结下来，收获主要几点：

- 第一次尝试编写通讯协议。原本以为 TCP/IP 是可靠数据传输，就可以放心大胆直接使用，不想还是出现了不少意外的情况，这启示我即使协议保证了运输层的可靠性，业务员还是需要从应用层再加以确认
- 充分理解了 P2P 连接的优劣。P2P 的调试耗费了我不少时间，由于不存在一个服务器导致必须进行双边调试，我在程序中各处加了不少输出信息，最后甚至使用 wireshark 进行抓包分析才得以成功
- 熟练了多线程编程。监听 socket、聊天收发消息都是依靠子线程完成的，本次大作业中我充分运用了多线程技术，也充分地遇到了各种各样的 bug，在解决的过程中熟练了异步多线程编程的技术

那么本次大作业到此也就全部结束了，最后还是要感谢贾老师一个学期的精彩讲课以及几位助教的答疑，没有你们的帮助我是不可能完成这样一份令自己满意的大作业的，感谢！

Reference

[1] https://stackoverflow.com/questions/17667903/python-socket-receive-large-amount-of-data

[2] https://www.cnblogs.com/lmg-jie/p/9629123.html

[3] https://github.com/Baidu-AIP/speech-demo/tree/master/rest-api-asr/python