# 计算机网络及应用大作业

# 基于中央定位服务器的 P2P 网络聊天系统


姓名：高子靖

学号：2017010917

班级：自 72


日期：2019 年 12 月 27 日

# 目录



## 1 需求分析与实现功能

本次作业旨在实现一个基于中央定位服务器的 P2P 网络聊天系统，也即通过中央定位服务器实现登入，登出与好友的 IP 查询等操作，在好友间的通信使用 P2P 来完成，具体见于下图：

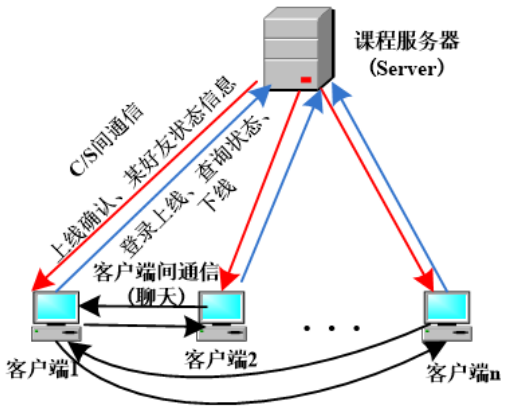



图 1: 系统说明

如上图所示，与服务器间进行 C-S 通信，使得账号能够完成上线，下线以及查询操作，客户端之间通过 P2P 实现通信，对于具体的需求，我所完成的功能如下所示：

**必做部分的所有功能我均实现：**

- 账号登陆上线/下线
- 维护通讯录，查询好友是否在线 (含聊天记录查询)
- P2P 文字通信 (Tcp)
- 文件传输 (10M 以上)
- 友好的用户界面

**选做部分我实现了以下的三种功能：**

- 群聊
- UDP 文字通信
- 图片传输与显示

接下来在报告中我将详细分析功能与设计方法，与实现的结果等一些关键的环节。

# 2 模块设计

## 2.1 整体流程与架构

首先我在整体上介绍我的聊天系统的架构，那么由于聊天的整体框架如上图 [1] 所示，下面我对于我所实现的功能进行一个整体上的介绍。

本次作业的设计仿照腾讯 QQ 的聊天流程以及界面进行设计，具体需要的流程如下所示：

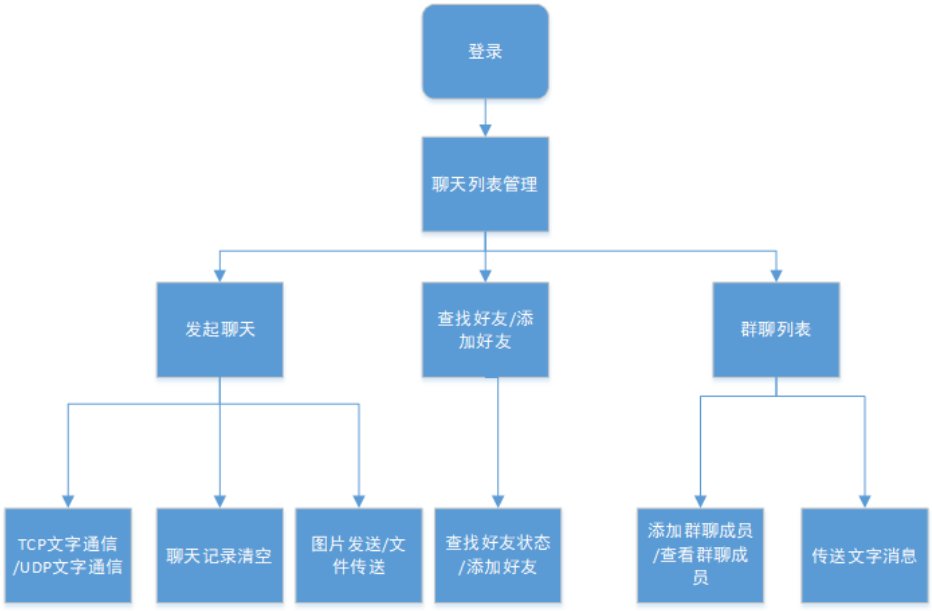



图 2: 程序整体流程

其中我按照了分模块调试的思路进行设计，因此在整体程序的完成过程中，Socket 编程的核心部分共设计了以下的几种必要的模块：

| 模块 | 功能简介 |
|---|---|
| CS 模块 (Client/Server) | 实现与服务器通信 |
| P2P 模块 | 实现基于 TCP 的文字通信 |
| UDP 模块 | 实现基于 UDP 的文字通信与校验等 |
| iMessage 模块 | 所有通信数据的传输协议 |

表 1: 模块名与简介

接下来我将对以上的四个核心模块进行介绍与分析，其中涉及到了一些多线程的使用等，具体的介绍将在详细设计中说明。这四个模块是本次作业在 socket 编程上的基础，是实现通信的核心设计。

## 2.2 数据协议

数据协议部分是整个通信流程的基础，其中包括了如何将必要的信息发送至对等方，对等方如何从获取的数据中” 解码” 得到正确的信息，均是以此数据协议作为基础。我的数据协议主要包括了如下的部分，在具体的实现时根据数据报的类型不同，数据格式稍有区别，并且在文本信息与文件信息包装成下方的数据报的方法也有所不同。整体数据报所包含的信息可见于下图：

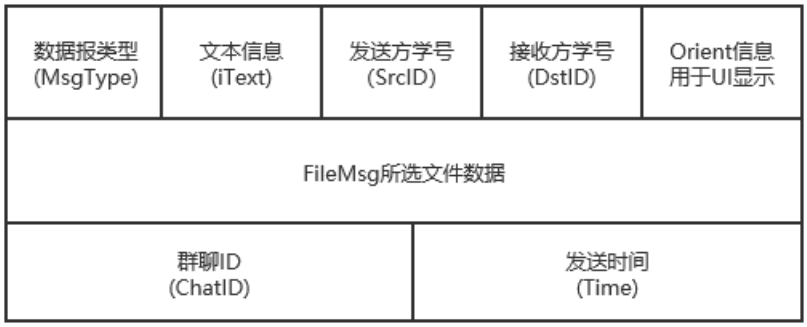



图 3: 数据报协议

上述的数据报便是我每次发送和接收信息得到的数据报，而将 String 以及 byte 数组等进行拼接，将数据报“编码”，以及接收到信息后如何将数据包“解码”，是我的数据协议主要实现的方法。

**Encode** 将数据报进行拼接的方法我重载了两种，分别针对于**仅包含文本信息的数据报**，和**包含文件信息的数据报**。对于前者而言，我实现的方法是将数据报中的每个字段加入 *String* 类型的分隔符，进而拼接成一个较大的 *String* 类型的数据报，将其整体转换成 *byte* 数组进行传输。而对于后者，由于文件的信息是以 *byte* 数组的格式读入的，因此不能将其转换为 String 之后进行拼接，这是因为无论以怎样的编码方式，不同扩展名的文件无法统一其编码方式，因此我改变上述的封装方式，将文本信息等按上述方法拼接好后转换为字节数组，并与文件数据信息进行字节数组的拼接。

**Decode** 下面介绍如何将字节数组格式的数据报进行解码。同样的，我对于两类数据报的解码方式是不同的。对于仅包含文本信息的数据报，这里将所有的字节数组直接转换为 *String* 类型，再根据预先加入的分隔字符串找出各个字段即可，返回为一个新的 iMessage 类。而对于包含文件信息的数据报，首先将其转换为 *String*，将其转换为仅包含文本新的数据重新进行拼接并转换为字节数组，这时可以获取文本信息的字节数组长度，那么剩下部分则全部为文件信息，以此获取文件信息。

此外，数据协议中的各个字段是聊天界面设计的基础，上述数据报仅包含需要传输的字段，其余字段可通过在对等方发送/接收时进行修改即可。我在聊天窗口的 listview 的数据模板绑定了我的 *iMessage* 类，因此其余 UI 的设计密切相关。

## 2.3 CS 模块

CS 模块也即 Client-server model, 实现了服务器客户端通信，本次作业中由于中央定位服务器由助教维护，因此在本地仅实现客户端的功能即可。具体而言上述的功能实现流程见于下图：

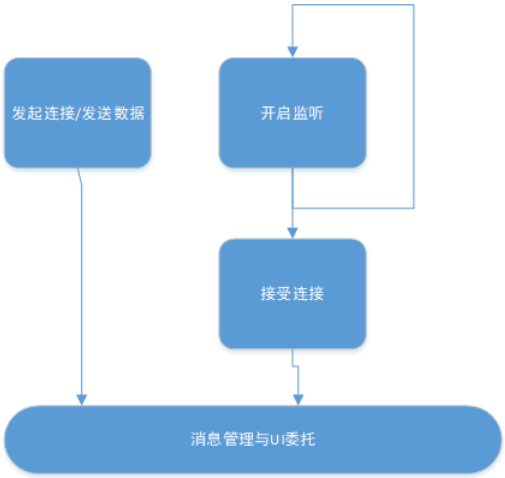

包含客户端向服务器端发送指令，得到服务器的响应，具体来说通信类型包括了以下的三种：

- 登录。向中央服务器发送学号 +"_net2019" 便可实现登录，服务器返还"lol" 说明已经登录成功。
- 查询好友状态，发送"q"+ 好友学号即可。返还“IP 地址”则说明在线，返回"n" 则说明不在线。
- 退出登录。发送"logout"+ 本人学号即可。返还“loo”则成功下线。

本次作业的 CS 通信通过 C# 的 Socket 类进行实现，是基于连接的 TCP 通信，也即首先与服务器建立 TCP 连接，在成功连接后按需发送上述的指令即可。

## 2.4 P2P 模块

本模块只要实现了在经过中央定位服务器查询后，对等方之间的通信。具体而言实现了文字通信，文件传输 (包含图片传输) 功能，具体而言 P2P 的实现流程如下所示：




上述反映的是 P2P 模块的状态机，即当进入到朋友列表的界面后，便在指定的某一端口开启 TCP 的监听，等待某指定端口上的 TCP 连接请求。而这里的监听需要使用 Thread 方法开启新的线程开进行等待，这样才不会阻塞主线程的运行。而 P2P 的实现也如上述的状态机所示，作为对等方间的通信，主要分为了两部分，也即**监听接收数据**和**发送数据**两部分。

开启监听部分我参照 c# 的官方例程使用 TCPlistener 与 TCPclient 进行实现，也即首先使用 Tcplistener 开启监听，在一个新的线程中通过无限循环的方式监听是否有连接挂起，也即使

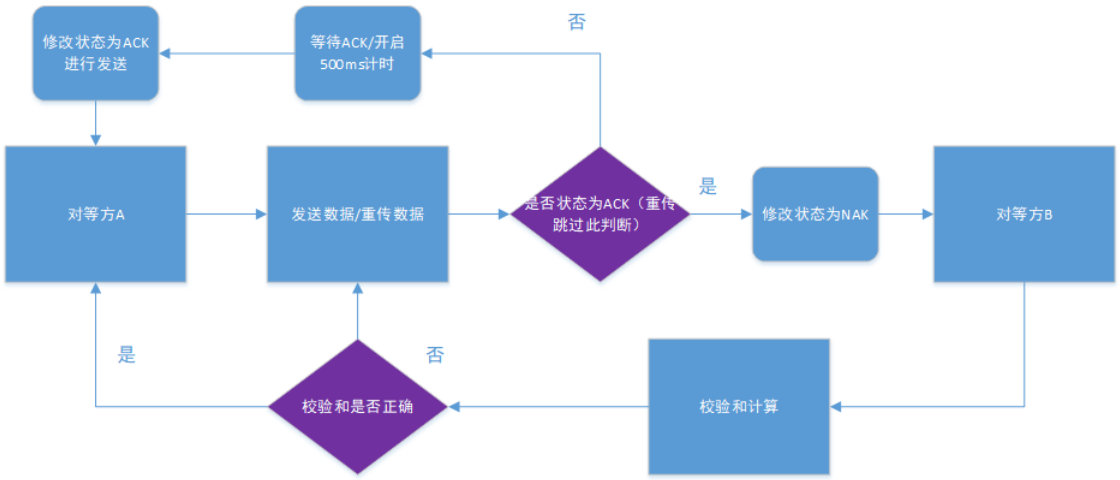
用了 TCPListener 的 *Pending*() 方法，此时进行相应的数据的读取，解封装，按照不同的数据类型进行不同的处理。由于这里需要首先绑定端口号开启监听，而 49152 到 65535 一般来说属于空闲端口号，因此我将端口绑定较大，为 53000.

需要说明的是，上述的数据处理是基于静态全局变量的，也即在传输层获取数据后，直接交由上层进行进一步的处理。实际上我的 P2P 模块包含了部分的上层处理，具体的介绍在第3节中给出。

发送数据的实现与登录时的 CS 架构比较相似，流程如图 [1] 所示，首先向中央服务器进行对等方的 IP 地址，获取对等方的 IP 后，使用 Tcpclient 尝试在对等方的端口建立 TCP 连接，在连接成功后使用流传输进行 write 即可完成数据的发送功能。

## 2.5 UDP 模块

UDP 模块不同于 TCP 之处在于 UDP 并非基于连接的，并且 UDP 不能保证数据的可靠传输。在本次实验中我实现了基于 UDP 的文字通信，为了数据的可靠传输，我加入了校验和，使用了基于 rdt2.0 的可靠数据传输方法。状态机如下所示：




以上的状态机主要基于 rdt2.0，由于测试中网络状态良好，因此 ACK/NAK 错误传输的几率也较低。因此我仅利用了当前状态是否为 ACK 来判断是否可以发送，仅有当 ACK 状态为真时才能够进行数据的发送，发送后立刻修改当前状态为 NAK 等待 ACK 的到来。在对等方收到数据中将会对所有数据进行校验和的检验，仅当校验和检验结果正确时回传 ACK，否则回传 NAK。当接收到 ACK 时，等待下一次的发送指令即可，若收到 NAK 则立刻重传数据，继续等待 ACK。

接收到其余的文字信息则交由上层进行数据管理，方法与 P2P 中类似，在这里不再赘述，在下一节中会详细介绍。

# 3 详细设计

在具有以上的几个模块的宏观上的介绍之后，下面我将按照我的实现顺序来逐一进行详细的分析。

## 3.1 必做部分

必做部分的实现内容由第一部分的任务需求中可见。

### 3.1.1 登录/登出

登录/登出操作通过客户端向服务器发起连接的请求并发送相应的指令即可完成, 状态机见于2.3, 在登录时通过独立的登录界面的 UI 设计。实现上也较为简单，仅完成 CS 模块的设计直接调用即可。

### 3.1.2 通讯录维护

通讯录维护涉及到了两部分内容，分别是在内存中的存储形式以及在本地的数据保存形式。在本次作业中我对于通讯录的维护也主要实现了两部分的内容：**好友列表以及群聊列表，不同好友及参与群聊的聊天记录维护**。当实现了上述的两方面的基础内容，才能够使用户有较为良好的体验，每次打开聊天界面后便会自动加在之前的聊天内容。但本维护功能仅限于本地，当更换设备后登陆同一账号无法进行聊天记录的同步。

首先说明我在内存中的通讯录维护方法，对于所有需要记录的内容我均使用了 *List* 来实现，主要基于以下的几个 List：

| List | 功能简介 |
|---|---|
| List<Groupchat> List_of_groups | 保存群聊列表 (包含群聊名称与群聊成员) |
| List<String> List_of_friend | 以“备注名（ID）”的格式保存 |
| List<iMessage> msg_list | 接受到的所有文字信息 (TCP 和 UDP) |
| List<iMessage> chat_msg_thisID | 所与某位同学聊天的聊天记录 |
| List<iMessage> group_msg | 某群聊的所有聊天记录 |

表 2: 通讯录维护 List

在上表中，前两个 list 即好友列表，在列表的 UI 中进行维护，通过添加/删除好友等操作改变其 List 内容，以字符串形式保存，而群聊则我定义了一个新的类 *Groupchat*, 其中封装了一个群组的群账号以及群组成员信息。下面的 *msg_list* 保存了通过 TCP 以及 UDP 接收到的所有文字类型的消息，也即需要保存到本地的消息，具体而言应该是还未被读取过的消息，若已经被读取，则此条消息则会被分配到相应的与某人或某群组的聊天记录中。而下面两个则是针对于某

一个对等方和某一个群组的聊天消息的记录，保存的形式均为我的数据协议类型 $iMessage$, 在从内存至本地保存时再进行进一步的切换。

而在向本地保存时则是通过保存在 bin 文件夹下以 txt 的形式进行保存。主要在本地保存以下四方面内容：

1. 好友通信列表。直接将好友的学号与备注信息按照行写入 txt 即可。在登录进入聊天列表界面时将 txt 读入内存并进行显示。

2. 好友聊天记录。在退出每一次的聊天框时进行保存，首先将 $iMessage$ 封装成一条 String 类型的消息，在打开聊天框时从本地读取相应的 txt 并且 Decode 成相应消息。

3. 群聊通信列表，我保存的格式为 txt，但由于一个群聊会包含不同的群组人数，因此我在每一个群聊前会首先写入一行“–BEGIN–A–GROUP–”作为分割符，下面一行记录群组 ID，再新起一行记录群聊人数，接下来每一行为群组成员的 ID。这样在读入的时候根据分隔符和人数能够初始化群聊列表。

4. 群聊聊天记录。这里的保存形式与好友聊天记录的相同，因为在每一条的聊天记录中会含有其群聊以及发送方接收方等重要信息。

此外，设计的部分交互功能如下所示：

**好友查找/添加** 点击聊天列表的查找，可以根据学号查找好友在线状态，合法的学号可以被添加至好友列表。

**删除好友** 在好友列表选中好友并点击删除便可从内容的列表中移除好友。

**群聊创建** 群聊的创建仅需输入群聊名称，创建后需要在其聊天框内进一步添加其他好友，这里会同步自己的通讯录以及被添加的好友的通讯录。

维护通讯录的流程如下所示：

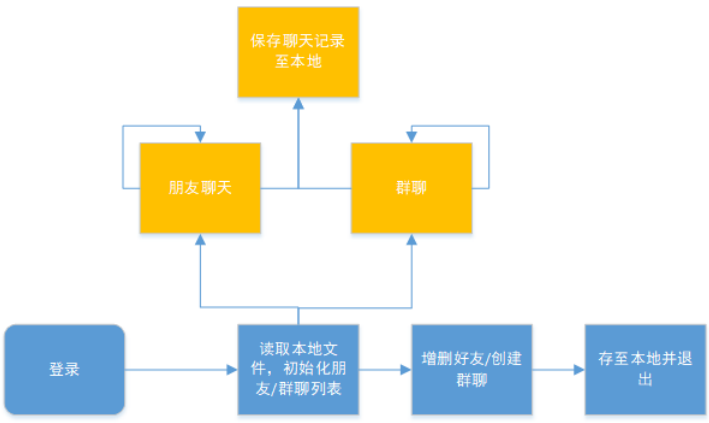



图 4: 通讯录维护

### 3.1.3 P2P 文字通信

P2P 文字通信是最为关键的部分，也是本次大作业中实现 P2P 网络聊天最为基本的内容，这里仅介绍实现 TCP 通信的方法设计，UDP 的部分见于选做的介绍。P2P 文字通信包含了基于 TCP 连接的数据传输以及接收到数据后的数据管理，由于我的聊天界面类似于 QQ，与每一个人的聊天界面会单独的弹出聊天窗口，因此流程如下所示：

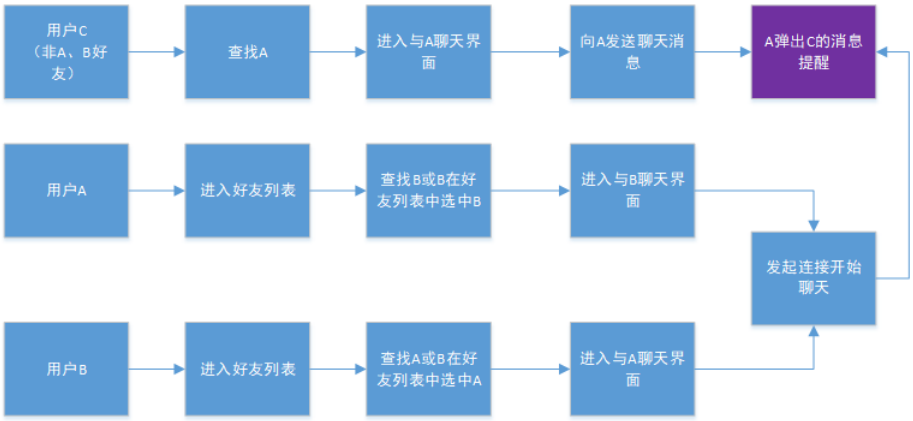



图 5: 聊天流程

上述包含了当双方都打开了和对方聊天界面的聊天情况和 A 并未打开与 C 的聊天框但受到 C 的消息的情况。当我进入到与某同学 D 的聊天界面后，便会开启新的线程，从 $msg\_list$ 中循环读取出当前同学发来的消息，并且将这一条消息从总消息列表中去除，加入到与当前同学的聊天记录中去。未被读取的内容仍暂被保存在上述的消息列表中，若直至退出程序仍为被读取，则将 $msg\_list$ 保存在本地，直至当用户打开那些未被读取消息的发送者的聊天框。

其中每一条消息包含的内容如下所示：

| 包含项 | 包含项 |
|---|---|
| 消息类型 | 发送者学号 |
| 接受者序号 | 发送时间 |
| 文本消息 | 对齐方向 (区分自己与对方) |

表 3: 文本消息

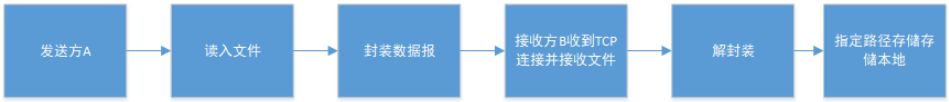
### 3.1.4 文件传输

文件传输部分基于文件部分的数据协议，为了使能够传输 10M 以上的文件，我设定了文件最大文件大小为 20MB，与文字通信不同的是，这里需要使用 FileStream 首先将文件的数据成字节数组的形式，在读取前首先判断文件的大小，通过文件的 info 便可以判断。理论上 Tcp 通信可以传送较大的文件如 1G 以上，但由于本题目中要求 10MB 以上的文件，因此我仅设置了

20MB 的 $maxfilesize$, 此外其封装数据时，将文件名称以及后缀名放入数据报的 $iText$ 字段中，可以使得接收方正确接收并按照相应的后缀名存储文件。

接收方在接收到文件后，首先将数据报解封装，具体的方法为首先按照文本信息解封装一次，可获得文件的字节数组长度，将文件信息与文本信息分开存放到不同字段，获取解封装后的数据报，则此时可对文件信息进一步保存，保存时设置后缀名与传输时相同。

在文件打开与保存时，我分别使用了 $OpenFileDialog$ 和 $SaveFileDialog$。需要由用户指定保存的路径而不是另存到默认路径。总体的通信方法与文字通信相类似，改变在于数据包所包含的内容，以及数据报类型，其简要的流程如下所示：




### 3.1.5 友好的用户界面

用户界面的设计我遵循了美观简洁与用户体验较好的两个原则，为了用户界面的美观，我在本次作业中使用了 C# 的开源控件库 $MaterialDesign$[1]，整体设计风格美观而简约。

整体上展示主要的一些界面，观感如下所示：

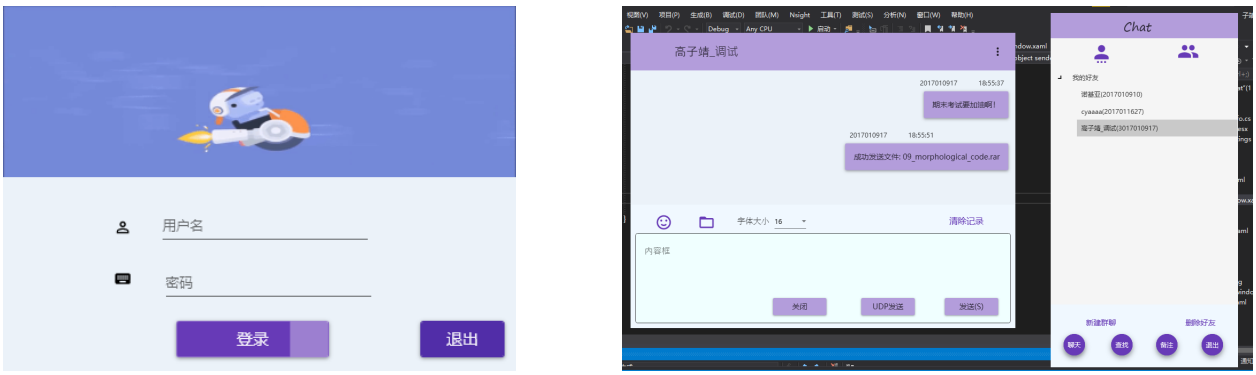



图 6: 登录界面与主窗体

可见主界面为发起任何聊天的接口，上方的两个选项选择显示双人聊天或是群聊，下方的按键实现了一些基本的通讯录管理与聊天发起等操作。

在界面上我花费了不少的时间，主要的难点在于聊天界面的设计以及聊天列表的显示，这两者我使用了同一个控件 $Treeview$。除此之外，例如登录界面的动态图以及整体的设计风格我均花费了一定的时间来设计，UI 在于用户交互上的流程基本按照了腾讯 QQ 的风格不同群聊以及好友在不同的窗体中进行聊天，查找好友修改备注等，都会弹出相应的窗体。下面我将叙述两个树形表的设计方法，分别是：聊天列表与消息列表。

**好友/群聊列表** 此列表由于仅需要保存好友的 ID 以及最基本的备注名，因此仅使用 Treeview 的 Item 属性便可以达到要求，因此在这里我便没有修改 Treeview 的 data 模板，直接使用

[1]一个 wpf 的控件库，详见 $material\ design\ in\ XAML$,http://materialdesigninxaml.net/

了 header 属性作为修改，所有好友添加后默认保存的格式为“学号 (学号)”，修改备注名后的格式为“备注 (学号)”，利用 selected 属性便可以获取当前选取的好友。而群聊列表则更为简单，在列表中显示的仅为群聊的 ID，因此在 UI 的显示上较为简单。

**聊天记录列表** 对于聊天记录，由于我为了实现类似于 QQ 那样的气泡效果，以及为了聊天信息显示的方便，我直接重新设置了 Treeview 的 Datatemplate, 整体上绑定了我的数据协议，直接通过数据协议中的各个字段绑定到其中就可以实现方便的消息管理。我主要竖直排列了方面内容，分别是收发人信息和消息。收发人消息通过 Text block 分别进行绑定，而下一层则是消息，消息我使用了一个紫色的 card 作为气泡，内部嵌入 textblock 并绑定数据报的 $iText$, 下方放置 Image 显示表情包，Image 的 source 绑定 FileMsg 即可，也即图片形式的文件字节数组形式可以直接作为 Image 的源进行显示。

对于聊天界面可以实现如下的 UI 效果：

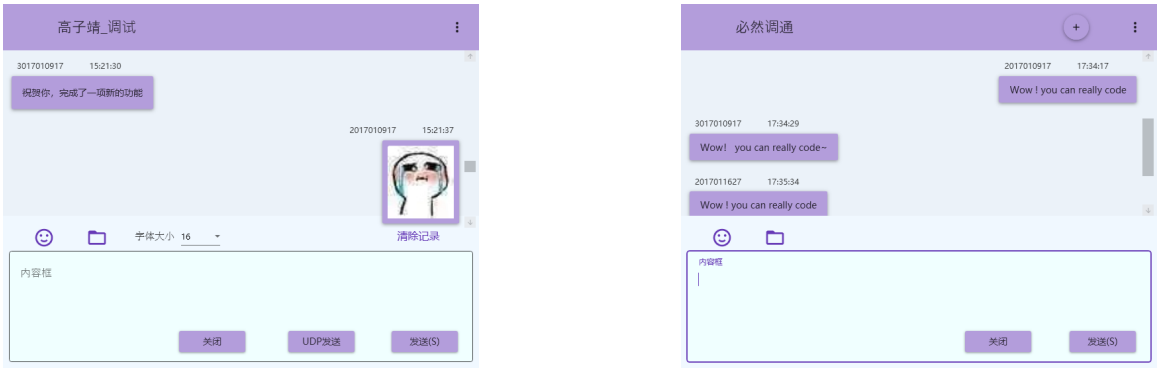



图 7: 聊天界面

具体的界面操作方法以及其他的部分 UI 见于第4节中的分析。其中的大部分 Icon 以及控件从 *materialdesigninxaml* 的 Demo 中[2]获取。

## 3.2 选做部分

### 3.2.1 群聊

群聊部分的实现有区别与双人聊天，但在传输层方面方法和双人聊天基本相同，都是基于 Tcp 的连接，然后向群聊中每一位非自己**并且在线**[3]的同学发送文字信息。此外，群聊的复杂之处在于群聊的管理，例如如何让参与群聊的同学得知自己被拉入了这个群聊等，因此我做了以下的管理方法。

1. 群聊创建。群聊创建时直接在本地的聊天列表创建即可，创建时需要输入群聊 ID，并且此 ID 一定是唯一的，若出现重复的 ID 则无法创建。创建之初群聊仅包含自己一人，需要进入到群聊的聊天界面中进一步动态添加好友至群聊中。

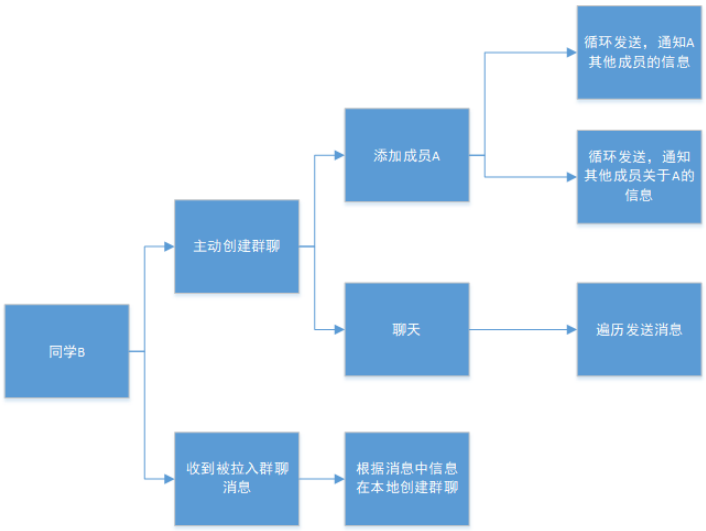
[2] https://github.com/MaterialDesignInXAML/MaterialDesignInXamlToolkit

[3] 由于这里仅能和在线的同学聊天，因此在线才能够收到消息

2. 群聊成员扩充，在聊天界面中有对应的学号输入框，输入正确的学号并点击上侧的加号，便可以将此同学加入到群聊中，这位同学同时将会接收到多条 MsgType 为 2 的消息，也即被添加消息，读取消息后该同学将会自动在本地创建该群聊，并且将陆续发送的成员信息添加到本地的群聊中。同时，在添加此名同学后，需要通知群组的其他各个成员添加此同学至其本地的群聊信息。至此，群成员的更新才算完成。

3. 群组聊天。这里的聊天由于每一次发送消息都需要首先向服务器查询各个同学的状态，因此之前会与中央服务器建立连接，而由于当群组成员较多的时候，需要此连接的保持，因此这里不便使用 UDP，因此在群聊的通信中我的文字发送均是基于 TCP 的，如上所示，遍历发送至每一位同学即可。

有关群聊创建及聊天的流程图如下所示：




除此之外，每一个聊天窗口构造时需传入其群聊信息，因此也可以动态更新的在右上角查看群聊群组成员的信息。此外需要说明的是，群聊的消息均包含 $ChatID$ 的字段，因此这是检查接收消息的基础，方便上层对数据的管理。

### 3.2.2 UDP 文字通信

UDP 文字通信的流程图已经在第2.5节中展示，在这里我首先说明 UDP 校验和方法。在本次的作业中，我在将文字信息打包好后，由于其 byte 是按照十进制每 8 位进行存储的，因此可以按照相同的方法，逐字节相加，并且每次相加后模 $0xffff$, 最终将此 10 进制数转为两字节，并且分别取高字节和低字节拼接为 16 位的校验和，进而取反，将校验和转为字符串并利用分隔符放入到发送的信息的 $iText$ 字段的最末尾，在收到 UDP 中对其 iText 进一步解封装即可。

解封装后应当计算数据部分的所有字节的校验和，并且与收到的校验和进行相加，若相加后为全 1(表现在字节为高低字节全为 255)，则说明校验无误，则发送 ACK 给发送方，发送方可以等待其下一次的发送。若出现了校验错误，则发送 NAK 至发送方，发送方收到 NAK 立刻重传

上次发送的信息，直至能够收到 ACK 为止，若始终重传无法收到 ACK，则在定时器耗尽时自动放弃此条消息，修改状态为 ACK，可以开启下一次发送。

整体上实现了类似于 16 位按位相加的校验和计算，经过测试网络状态较好，发送方总能收到 ACK，同时收到 ACK 也能说明了上述的校验方法的正确性。

### 3.2.3 图片传输与显示

图片传输显示实际上是一个更加偏向 UI 的设计，其本质上属于文件传输，与文件传输的区别在于上层对于数据的处理不同，由于图片用于在聊天界面保存表情，因此在发送时将 Image 绑定读取的数据的字节文件即可实现显示，在接收到图片类型的消息时，解封装后相应显示在 UI 上，但不再将数据保存至本地，因此可以实现聊天过程中的图片以及表情的功能。

## 4 结果与分析

在本节中，我将详细说明并展示我所实现的各种功能。按照操作的顺序进行展示。

**登录界面** 下方为登录界面，登录界面包含了动态背景图片及登录账号以及密码等。

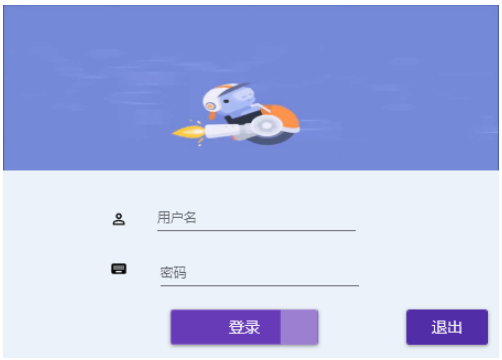



图 8: 登录功能

**通讯录管理** 在通讯录管理中，首先如下图 [12] 所示，分别展示了群聊列表和好友列表，在好友列表中可以看到好友的备注名以及其学号，在最下层有一排功能键，其共同构成了通讯录维护的功能。选中好友后点击【备注】可以更改备注，点击【删除好友】可以将此好友从本地删除。而点击【聊天】可以开启和当前好友的聊天，点击【查找】则会进入一个新的对话框，其中可以按照学号查找好友，并返回该好友的状态，并且可以通过此界面添加新的好友。【退出】键则会向中央定位服务器发送退出的指令，使得账号在服务器上下线。【新建群聊】会使得本地创建一个新的群聊。

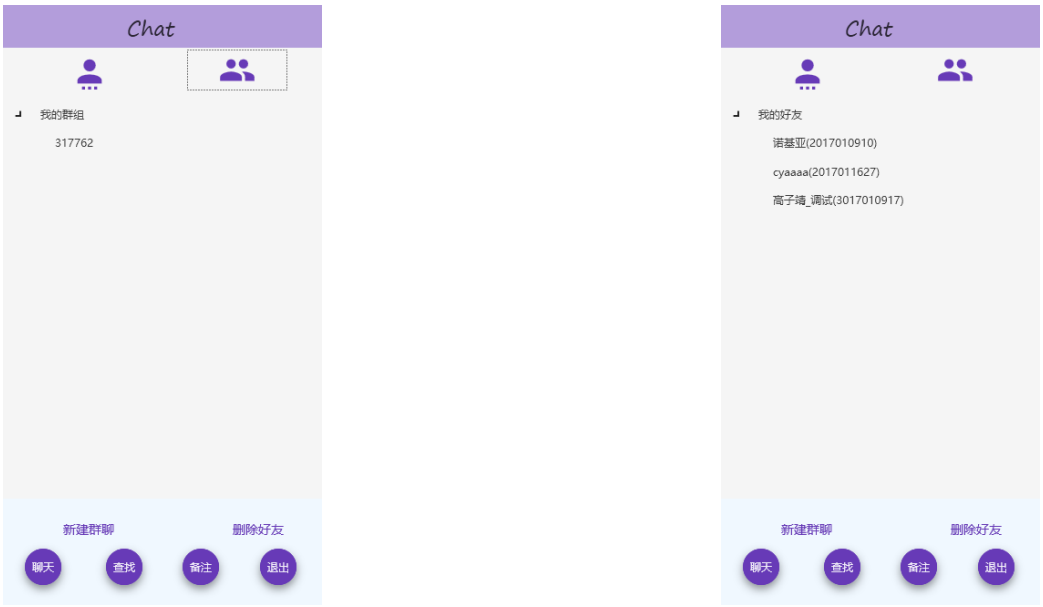



图 9: 通讯录管理

**好友查找** 与 **创建群聊** 实现的界面如下所示, 左图为查找好友状态，则会返回显示当前的好友状态，包括会提箱好友学号是否在课程范围内等，右侧则是输入群聊 ID 创建群聊。


图 10: 好友查找与创建群聊

**双人聊天** 与 **群组聊天**


图 11: 双人聊天与群聊

在上述的左图中展示了双人聊天以及图片的传输和显示，右侧为三人聊天的群聊功能 (由于

借到三台电脑比较困难，因此在本张截图之后对 UI 做过微小的调整)。可见已经实现了基本的功能，在左图中也包含了 UDP 传送与发送 (TCP 发送)，其起到的效果都是相同的。

**文件传输** 我选取了一个大小为 10.2MB 的文件进行传输，得到以下的效果，这里由于我直接使用 3017010917 在本地 IP 进行登录，因此测试时文件会被发到本地，经过测试发往其他设备也不存在问题。经检验得到的数据保存后没有损坏且大小与发送前相同。


图 12: 文件传输

**群组聊天成员查看** 我在调试时创建了一个包含两名成员的群组查看方法如下所示：


图 13: 群聊成员查看

点击右上角的扩展槽，可以看到群聊的成员信息，但值得注意的是，群聊中的文件传输键和表情键没有相应的作用，群聊实现的功能仅为群组成员间的文字通信。

以上便是我所实现的基本功能，具体的操作说明见于同目录下的 README.pdf。

# 5 出现的问题及解决方案

本次作业是我第一次接触 Socket 编程，整体上我按照功能进行调试，由于和其他学科的大作业并行，因此我一般每隔两三天会调试其中的一个功能，因此完成大作业的周期也比较长。在大作业的完成过程中我出现了许多的问题，尤其是在起初面对 P2P 通信时，由于对线程的使用

方法以及 C# 提供的 socket 编程方法的不熟悉，我画了很长时间阅读博客和官方文档。中途也出现了很多的 bug，下面我总结如下：

1. 在开始阶段，我本想使用 c# 的 Xamarin 进行跨平台的 IOS 编程，但是由于 IOS 的限制，需要有一台 MAC 来做远程编译，另需一台 Windows 完成编程。起初阶段在 Mac 上配置 visual studio 的 Xamarin 环境以及将 windows 与 mac 进行远程连接，都会出现问题，windows 总是难以在 ios 编程环境下连接到 Mac，考虑到后期调试的会比较难以实现并且起初阶段的硬件配备不足，我只好放弃了移动端的想法，转而选择 c# 来完成。

2. 线程开启方法。在刚刚要开启 P2P 的阶段，我对于如何发送和接收管理消息没有实现的思路，当通过官网了解到其 Tcplistener 开启的方法以及接收数据的方法后，我对于线程的管理还不是很熟悉，总是会出现程序退出，线程仍在运行的情况，或者由于线程的开启问题等等阻塞了主线程，在经过一番各种细节上的修改后，我终于能够正常的开启监听，并且接收消息。

3. 文件传输时数据协议的修改。因为我先做出了文字通信，但是文字通信我使用的协议仅能够满足文字通信，这是因为我将其按照 String 的分割符进行拼接的，但是当我将文件的数据同样转成 String 进行拼接，则收到文件后总会出现文字损坏或者出现空白的情况，我使用了 Unicode,UTF-8 等各种编码方式，但是任何一种编码都无法保证在不同的后缀名传输后保证文件的质量，因此我选择直接重载数据的封装方式，直接将文字信息转为 byte 后与文件数据进行拼接，这样一定不会损坏问价，但是修改数据协议对于整个程序的影响较大。会将一系列的 bug 修改后才可以正常运行。

除此之外，调试过程中也遇到了许多其他的问题，例如两台设备对话时由于防火墙导致” 目标计算机积极拒绝”, 以及当我开的端口号较小时，对方可能因为端口被占用而无法收到消息，整个完成的过程中出现了各种各样的错误，但是每一个错误都是我后期调试的经验，当我在后期设计 UDP 时，便更加顺利。

# 6 总结

如上所说，我在完成作业的过程中遇到了许许多多的错误，这次的大作业也是我在本学期内完成的工程量最大的大作业。不过，通过这次大作业，使我学会并且掌握了 Socket 编程的方法，对于传输层协议以及可靠数据传输等理论知识的掌握和理解都更加的深入。我觉得虽然过程中遇到了很多问题，但是每完成一个新的功能都能够带给自己成就感，给自己继续做下一个功能的动力。也正是通过这次的大作业，通过对其 UI 界面的设计，使我对于 WPF 各种控件的功能与使用方法更加熟悉，包括了各种动态绑定以及重写 datatemplate 等，我想这对我之后的其他 UI 设计都是有帮助的。总而言之，本次作业将课上的理论知识与应用相结合，让我通过课外的方法查阅并学习了很多知识，获得了很大的收获。

# 7 参考文献

[1] https://docs.microsoft.com

[2] https://www.cnblogs.com/zh7791/p/9549542.html