基于中央定位服务器的 P2P 网络聊天系统设计

实验报告

姓名：赵文亮

学号：2016011452

日期：2019 年 1 月 6 日

目录

1 需求分析

本次实验中，我将实现一个基于中央定位服务器的 P2P 网络聊天系统，该系统的主体结构如图 1 所示。用户通过服务器来登录、登出、查询好友的 IP 地址，而整个通信的过程是用户之间实现的，不需要服务器参与。

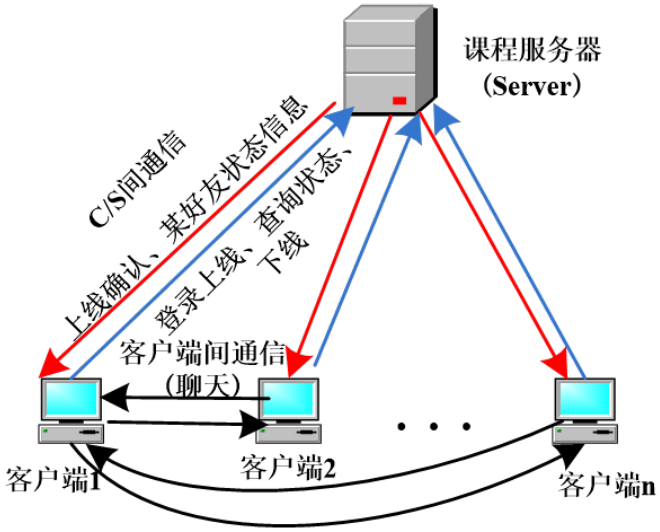


图 1: 系统结构图

具体来说，本次实验中实现了如下功能：

- 必做部分：
  1. 账号登录/下线
  2. 通讯录（联系人的查找、增加、删除），好友在线状态查询
  3. 好友之间的 P2P 通信
  4. 文件传输（10MB 以上）
  5. 友好的用户界面（仿腾讯 QQ）
- 选做部分：
  1. 群聊功能（一对多通信）
  2. 聊天记录查询
  3. 语音发送
  4. 发送 Emoji 表情与动态表情
  5. 拍照并发送图片

本次实验我使用 Java 语言，基于 Android 系统编写了手机客户端。

2 总体设计

本节中将介绍整个系统的总体设计，包括系统运行的流程、数据结构的设计和各个功能的实现思路。

2.1 整体流程

程序整体的流程如图 2 所示。程序初始界面为登录界面，登录后进入到主界面。主界面中包含三个标签页，分别是最近消息管理、联系人管理和设置。在联系人管理标签页中，用户可以进行联系人的增加、删除和查找；在设置标签页中，用户可以清空联系人或消息列表；在联系人管理和最近消息管理列表中，用户点击联系人或最近消息可以发起聊天，进入到聊天界面。在聊天界面里，用户之间可以进行多种形式的信息交互，并可以进入到发起群聊界面，邀请其他好友加入群聊。

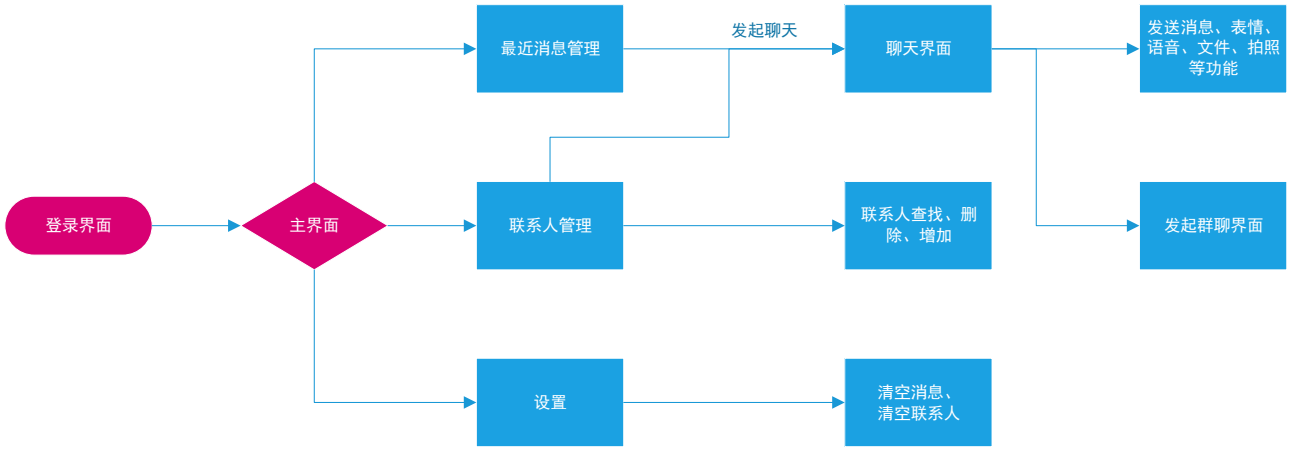


图 2: 程序整体流程

2.1.1 双人聊天流程

首先我们来分析两个用户之间聊天的流程，设两个用户分别为用户 1 和用户 2，则从登录开始的一个完整的流程如图 3 。可以看到，由于整个系统为 P2P 的架构，两个用户的行为完全对称。下面从用户 1 的视角来介绍该流程。用户 1 登录后，进入到主界面，此时用户 1 已经开启了服务器端[1]，等待接受连接。用户 1 在联系人标签页中添加用户 2 为好友并发起聊天，此时用户 1 进入到了聊天界面。用户 1 首先向中央服务器查询用户 2 的 IP，若 IP 存在，则向用户 2 发起一个 TCP 连接。连接建立后，用户 1 便可以向用户 2 发送消息。

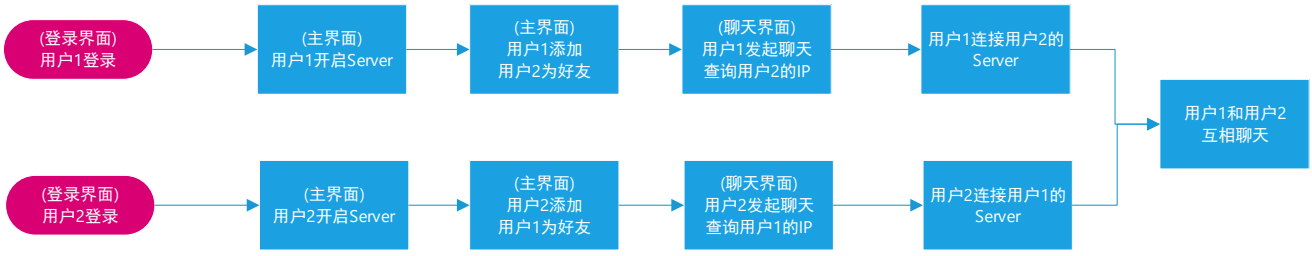


图 3: 两个用户聊天流程图

在 P2P 的架构中，每个用户同时是服务器和客户端，所以设计较为复杂。具体的协议和状态机将在第 3 节给出。

[1]本文中的“服务器”（或 Server）特指 P2P 用户开启的服务器端程序，“中央服务器”特指中央主楼内用于定位的主机

2.2 多人聊天流程

多人聊天的过程与双人聊天类似。例如，若一个群组中包含用户 $1 \sim n$，此时用户 1 想要发送一条消息，事实上只需用户 1 向其他 $n-1$ 名用户发起连接，并分别发送消息，也就是可以看成 $n-1$ 次的双人聊天。所以一旦实现了双人聊天的功能，多人聊天的功能也并不困难。

2.3 数据结构设计

本程序中涉及到的数据结构设计主要包括三种：最近消息、联系人、聊天消息。需要指出的是，这些数据结构中的一些字段只是为了设计的完整性，在数据结构设计阶段引入的，并不一定在最终的程序中使用[2]。

2.3.1 最近消息

最近消息中包含以下几个字段：

- 学号
- 姓名
- 时间
- 未读消息数
- 上一条消息

2.3.2 联系人

联系人中包含以下几个字段：

- 学号
- 姓名
- 在线状态

2.3.3 聊天消息

聊天消息中包含以下几个字段：

- 发送方学号
- 接受方学号
- 消息类型，包括命令、文字、表情、图片、文件、语音（后文记作 CMD，TEXT，EMO，IMG，FILE，AUDIO）
- 消息状态，包括发送中、已发送、发送失败
- 发送时间
- 发送进度（图片、文件、语音）

[2]本学期时间有限，故本程序更加关心核心的部分

- 文件长度（图片、文件、语音）
- 语音长度（语音）

其中六种消息类型中，后三种的本质都是文件，所以它们应该具有发送进度和文件长度的属性；在发送语音时，我们往往希望知道语音的长度，所以我为语音类型的消息添加了语音长度的字段。

2.4 功能实现思路

2.4.1 网络连接

本程序中所有的网络连接都使用 TCP，目的是使连接更加可靠。与中央服务器之间的连接和消息发送都可以通过返回值判断本次操作的成功与否，P2P 用户之间的通信也将通过应答信号来保证连接的可靠。

2.4.2 数据管理

我使用 Android 中的 Room[1] 数据库完成了最近消息和联系人的数据管理。Room 数据库提供了 SQLite 数据库上层的更加稳定的数据库访问，是目前 Android 官方推荐的数据库解决方案。这两部分数据保存在用户本机上。

而对于聊天消息，我使用 Bmob 后端云 [2] 来完成数据的管理。将聊天消息保存在云端主要是考虑到用户在聊天的过程中可能会产生大量的聊天消息，将这些消息都保存在本机会占用大量的手机内部存储空间；另一方面，如果聊天消息都在本机保存，那么为了能够查看过去的消息记录，同一条消息需要保存在多个用户的手机上。例如，一条在 10 个人的群组中发送的消息就要分别在这 10 部手机上占用空间。综合考虑后，我决定使用云端数据库来存储聊天消息。在聊天的过程中，所有的聊天内容都通过 P2P 的方式发送。当接收方收到消息后，会检查 Bmob 后端数据库中是否已经存在本条消息，如果不存在则将此条消息保存在云端（避免群聊时重复保存）。

2.4.3 消息传送

聊天消息的传送主要分为两大类：文本与文件。文本的传送比较简单，只需要固定数据报格式，使用 TCP 发送一个字符串即可。文件的传送（包括文件、图片和语音）则有所不同。首先，需要一个前导数据报来告知接收方文件的类型、大小等信息，再将文件拆分逐块发送；接收方再根据文件的大小逐块接受文件。对于文件类型的消息，我将文件本身保存在用户本地，而将前导数据报保存在云端消息数据库中。

3 详细设计

本节将分功能来详细分析设计的思路。

3.1 必做部分

3.1.1 账号登录/下线

中央服务器的指令集如表 1 所示。程序开始时，首先向中央服务器发起连接（IP 地址：166.111.140.14，端口：8000），当用户点击登录按钮时，将用户的学号和密码按照指令集的格式发送给中央服务器进行验证，验证成功后跳转到主界面。

表 1: 中央服务器指令集

通信类型	客户端发送指令	服务器返还指令
登录	用户名：本人学号，密码：net2018。 例：2016011452_net2018	“lol”
查询好友状态	”q + 好友学号”。 例：q2016011451	IP 地址（在线） “n”（不在线）
下线	”logout+ 本人学号” 例：”logout2016011452”	“loo”

3.1.2 通讯录（联系人的增加、查找、删除），好友在线状态查询

通讯录维护 我使用 Room 数据库来维护通讯录和最近消息两个标签页中的数据，[3] 中介绍了一种使用 Room 数据库、LiveData 和 ViewModel 的解决方案。例如，为了显示联系人列表，我需要建立以下几个类：

1. Contact：联系人的实体类
2. ContactDao：DAO（Data Access Object，数据获取对象）
3. ContactReository：进一步封装，适用于多种数据源
4. ContactViewModel：用于数据库与 UI 之间通信
5. ContactListAdapter：将数据显示在列表中，Adapter 是 Android 编程中的一种常见的对象，用于将数据显示在控件上

虽然这个过程十分复杂，但是这样的操作使得数据库的操作和 UI 的操作很好地分离，可扩展性、封装性、稳定性都较好。

联系人查找 我在主界面顶部设计了一个搜索框，当用户在其中输入学号时，程序会时时查找当前联系人列表中是否存在输入内容的匹配，并实时更新过滤结果，由此实现了联系人的查找。

联系人增加 在搜索框中输入一个学号，若该联系人不存在，则联系人列表显示为空。此时可以点击顶部栏右侧的加号来增加联系人。当实验中我发现，如果一个学号从未登录过，那么向中央服务器发送“q+ 学号”的查询命令就会导致中央服务器超时而没有返回结果。所以，用户点击加号时，程序首先使用正则表达式来判断学号格式，再向中央服务器发送“q+ 学号”命令查找该学号是否存在。如果学号格式不对或者改学号从未登录，则认为本次添加无效；否则成功添加。

联系人删除 用户长按列表中的某一联系人，即可将其删除。这一步骤通过调用数据库的相关方法即可完成。

好友在线状态查询 联系人标签页初始化时，通过向服务器发送“q+ 学号”命令逐个查询列表中的好友，并使用正则表达式判断返回值是否为 IPV4 格式，从而判断好友是否在线。此外，我还基于 SwipeRefreshLayout 设计了下拉刷新的功能 [4]。

3.1.3 好友之间的 P2P 通信

数据包格式 聊天消息类 ChatMessage 中包含的成员见第 2.3.3 ，数据包设计的原则就是将一些必要的成员编码为字符串，这样便可以很简单地完成聊天消息的发送。按照从前到后的顺序，编码为字符串后的一个聊天消息中包含的内容如下：

1. 数据包长度
2. 发送方学号
3. 接收方学号
4. 消息类型（CMD：0，TEXT：1，EMO：2，IMG：3，FILE：4，AUDIO：5）
5. 时间字符串的长度
6. 时间字符串
7. 消息内容

有了这个规则，便可以对一个消息进行编码和解码。

服务器端与客户端的交互 P2P 服务器端的状态机如图 4 所示。用户进入到主界面后，服务器即开启，进入监听模式（本程序中监听 50002 端口）。每当有客户端发起连接，服务器都会响应该连接并为其新建一个线程来接受消息，随后继续监听。这样的设计保证了用户接受消息的过程中不会错过其他用户向他发送的消息，也是比较合理的设计。

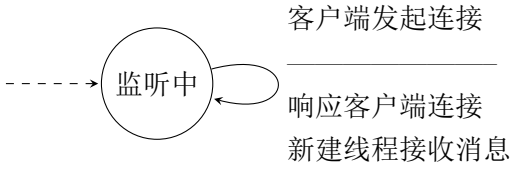


图 4: P2P 通信服务器端状态机

假设用户 1 向用户 2 发起会话，首先要保证用户 2 已经登录（即服务器已打开）。用户首先获得用户 2 的 IP，再去连接用户 2 的 50002 端口。用户 2 的服务器端响应该连接，并开启新的通信线程。此时，用户 1 便可以向用户 2 发送消息。以文本消息为例，首先将该消息的相关信息保存在 ChatMessage 的对应属性中，再将其编码为字符串发送给用户 2 的服务器端；用户 2 收到后，会返回一个“ACK”应答信号表示收到。当用户 1 退出聊天界面时，会向用户 2 发送一个类型为 CMD 的消息，其内容为“BYE”，表示退出；用户 2 收到后，将该消息线程退出。

3.1.4 文件传输（10MB 以上）

与文本传输不同，文件传输不能够将文件一次性发送，而是应该每次读取一小块数据并发送，接收方每次收到一小块数据并保存。发送方首先创建一个引导消息，其类型为 FILE，内容中包含了文件的本地路径和文件长度，二者使用了 “?” 符号进行分隔[3]。接收方收到前导消息后，根据消息类型得知接下来要接收文件，并根据消息内容得知文件的路径和长度，准备接收文件，并根据文件名新建一个空文件[4]。

[3]文件路径和文件长度中不可能有 “?” 符号，所以这样的分隔是安全的

[4]默认路径为：系统存储目录/johnwilliams/qq/file/

发送方此时开始读取文件，每次读取 1024 个字节并发送，直到发送完毕；接收方每次接收 1024 个字节并写入文件，直到已接收字节等于文件长度。无论是发送方还是接收方，都可以实施计算文件传送或接受的进度，进而更新界面中进度条。

文件传送后，点击文件即可选择默认应用打开，这一设计十分的人性化。

3.1.5 友好的用户界面（仿腾讯 QQ）

本程序的用户界面仿照腾讯 QQ[5] 设计，为了使界面更加友好，我在 UI 设计和交互体验上面用了很多心思。然而本文篇幅有限，过多介绍界面相关的内容略显喧宾夺主。程序中所有的界面在第 4 节中都将展示出来，本节中仅介绍一些 Android 开发中 UI 设计的技巧。

Selector 在手机 App 设计中，我们常常需要在用户按下一个按钮的时候改变该按钮的样式，如图 5a 所示，这在 Android 中可以通过 Selector 来实现。首先准备两张图片，例如图 5b 和图 5c 。再编写 selector 的 xml 文件，其中指定正常状态的图片和选中状态下的图片即可。

(a) 效果图

(b) 正常时 (c) 按下时

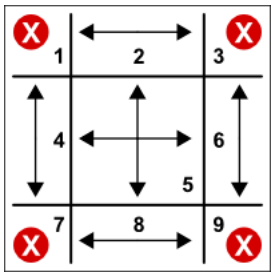
图 5: Selector 效果示意图

9-patch 一些情况下，当我们为控件设置背景图时，我们常常希望控件的背景能够随着控件的大小而自适应改变。然而对于那些边缘不是方形的背景图，直接进行缩放效果并不好。以聊天中的气泡为例，气泡的四角都是圆角，还有一个指向消息发送者的箭头。我们希望内容大小变化时，仅仅是气泡中央的区域进行缩放，而边缘保持不变，这就需要使用 9-patch 位图，其示意图如图 6a 所示。其中 1、3、7、9 四个区域的大小始终保持不变。在 Android Studio 中，我们可以手动设置图 6a 中的各个边界，并可以指定内容的显示范围。图 6b 展示了聊天中的气泡，可见它能够很好第适应不同长度的内容，这都是 9-patch 位图的功劳。

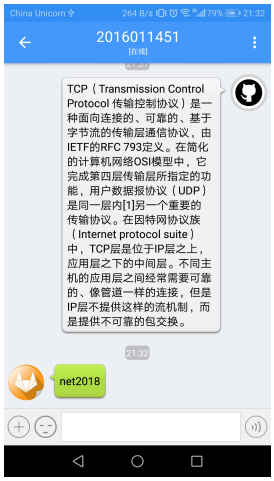
Animation 微信和 QQ 在播放语音时，语音气泡上会有一个动画，用来显示现在正在播放语音。这个效果的实现方式如下：首先准备图片，对应着动画中的每一帧；再编写 animation-list 的 xml 文件，其中指定每一帧使用的图片和该帧的持续时间。

图片的获取与处理 程序中很多地方都用到了各式各样的图片，这些图片的获取和处理其实比较困难。本程序主要有以下几个图片来源：

- ICONFINDER 网站 [6]
- Github 上的开源项目 [7]



(a) 示意图



(b) 效果图

图 6: 9-patch 示意图与效果图

- 将 QQ 安装包解压，可以得到 QQ 中的所有图片

当然，获取了图片之后还需要进行处理。我们希望图片的背景透明、颜色符合 App 的主色调，这些都可以在 PS 中完成。

3.2 选做部分

3.2.1 聊天记录查询

聊天记录的保存基于 Bmob 后端云 [2] 完成。Bmob 后端云提供了一套完成的 API，可以对数据库进行各种操作。用户在收到一条消息后，调用 Bmob 的查询语句检索数据库中是否已经存在此条信息（防止群聊时重复保存），如果没有检索到，则将此条信息保存到数据库中。

用户每次点开聊天界面时，向 Bmob 数据库中查找满足条件的消息，并显示在消息列表中。

3.2.2 语音发送

语音发送主要包括语音录制和语音发送和语音播放三个部分。我使用 Android 中的 MediaRecorder[8] 完成了语音录制部分。MediaRecorder 可以设置音频输出格式、编码方式、保存路径、采样率等诸多参数，并可以测量声音的大小。我使用一个心跳包定时获取 MediaRecorder 检测的音量，并在界面中实时显示出来（详见第 4 节）。

录音结束后，发送语音的本质其实就是发送文件。接收方收到语音文件后，点击即可播放。播放语音的功能可以使用 MediaPlayer[9] 来实现。

3.2.3 发送 Emoji 表情与动态表情

Emoji 表情和动态表情都是事先保存在资源文件中的（随 apk 打包），所以发送表情的实质是指定表情的编码。通常的 Emoji 使用 Unicode 编码，以`\ue` 开头。发送 Emoji 时，只需要将相应的编码作为文本发出，接收方收到后使用正则表达式匹配字符串，找到其中所有的 Emoji 编码，并将对应区域替换为资源中的 Emoji 图片即可。

动态表情的发送原理类似，本程序中导入了 16 张蜜桃猫表情包，并使用 `\mt` 作为编码的开头。发送 GIF 表情包时，消息类型应该设置为 EMO，接收方判断消息类型为 EMO 后，使用 GIFImageView[10] 显示动态表情包。

3.2.4 拍照并发送图片

拍照的功能通过调用系统的相机来实现 [11]。将预先设定的路径传给拍照的 Intent，即可将拍照后的结果保存在该路径中。随后即可以调用发送文件的方法，将照片传送出去。

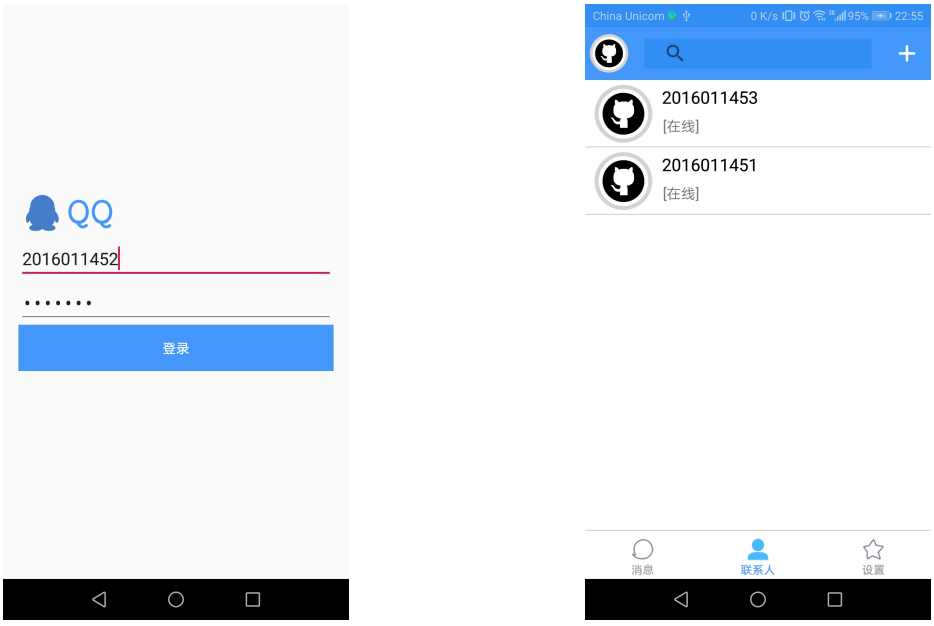
3.2.5 群聊功能（一对多通信）

群聊功能首先应该从创建群聊开始。两个用户聊天时，用户可以点击菜单栏中的发起群聊邀请其他好友加入群聊。设置群聊名称后，发起群聊者（默认为群主）向所有群成员发送一条类型为 CMD，内容为所有群成员学号和群名的消息。其中群成员的学号与群名之间通过 “$” 符号来分隔，不同群成员的学号之间通过 “;” 来分隔。当群成员收到这个消息时，会在自己的通讯录中加入这个群聊的信息。

群聊的过程中，每位群成员都将建立与其他所有在线群成员的连接。每次发送消息时也遍历除自己以外所有的成员逐一发送。其本质与两人通信并无差异。

4 结果分析

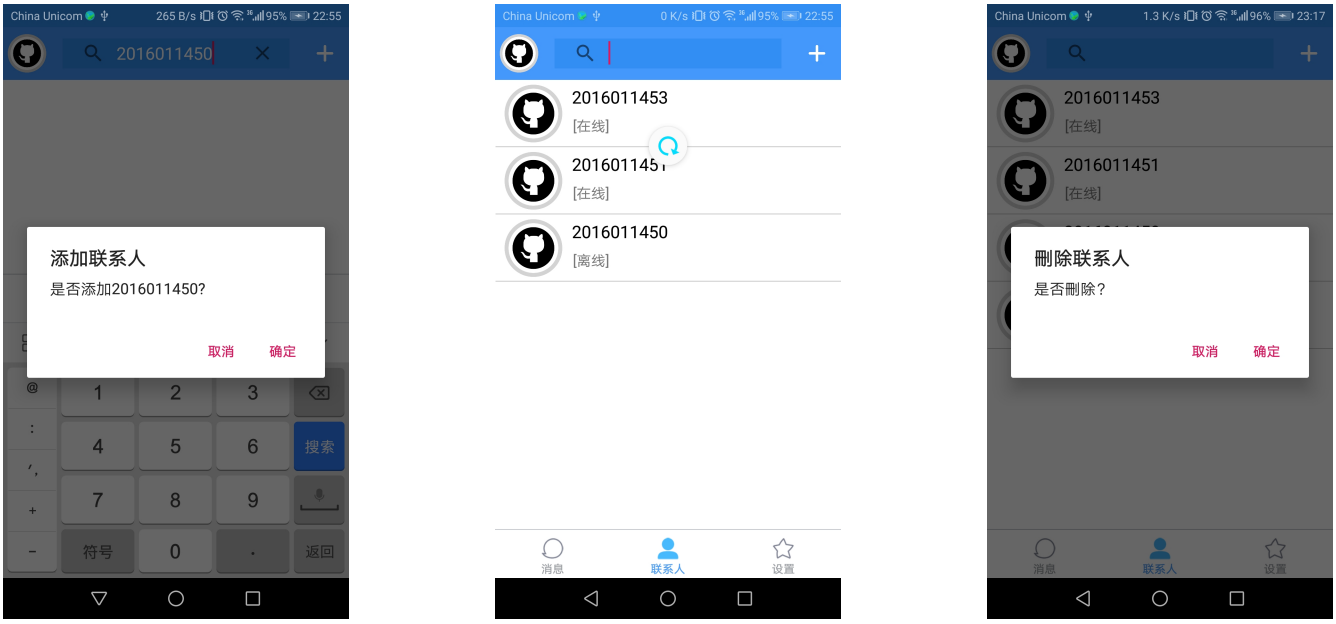
本程序的运行环境为 Android 5.0-Android 9.0。我将程序的安装包放在了我的服务器上，可以访问`http://www.johnwilliams.online/MyQQ.apk`下载。为了更好地展示本程序的结果，我简单录制并剪辑了一个视频，展示了本程序的主要功能，见 demo 文件夹。本节中将展示程序运行结果的一些截图。



(a) 登录界面 (b) 主界面

图 7: 登录界面与主界面

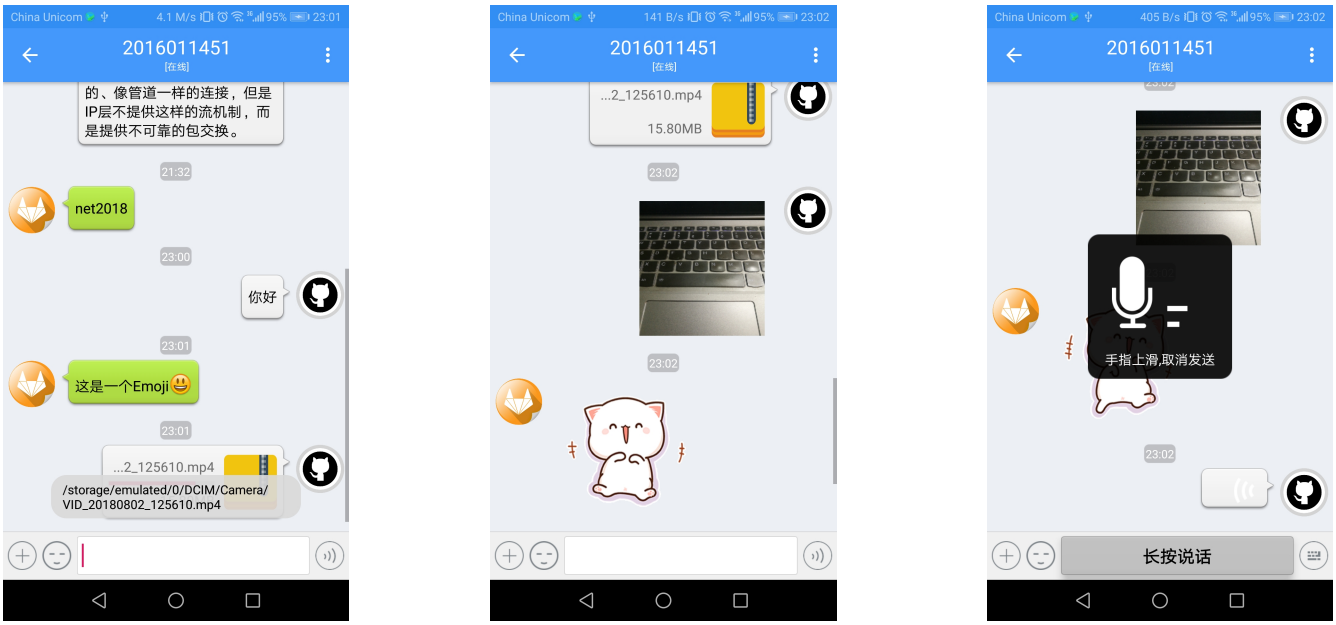
图 7 展示了登录界面与主界面（此时为联系人标签页），图 8 展示了对联系人的操作，包括添加联系人、刷新联系人在线状态、删除联系人。



(a) 添加联系人 (b) 刷新联系人 (c) 删除联系人

图 8: 联系人操作

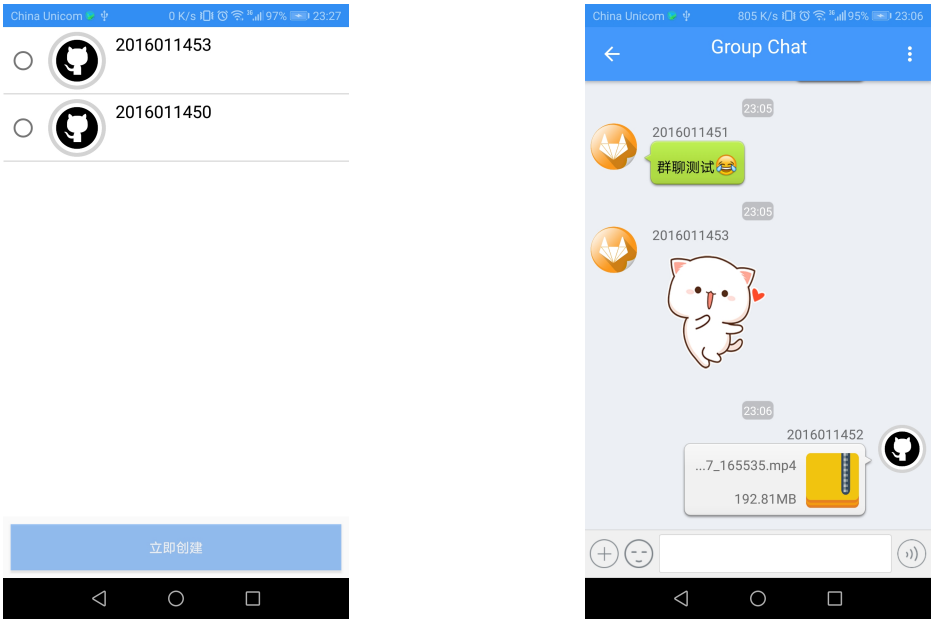
图 9 展示了 P2P 通信中发送文本、Emoji 表情、文件、图片、动态表情、语音的过程。



(a) 文本、Emoji 表情、文件 (b) 图片、动态表情 (c) 语音

图 9: P2P 通信过程截图

图 10 展示了创建群聊和群聊中发送消息的过程。其中成功发送了一个将近 200MB 的文件，可见程序的鲁棒性很好。



(a) 创建群聊 (b) 群聊通信

图 10: 群聊过程截图

5 总结

本次实验中，我第一次编写了网络应用程序，也是第一次编写 P2P 程序。不得不说，P2P 程序真的很难调试。由于所有用户的对等性，我不得不分开调试。最初调试与中央服务器通信的部分时，我使用 Android Studio 的虚拟机运行程序，并用 Wireshark 抓包来确认我是否成功发送和接收数据。一端调通后，再用调通的程序来调试没调通的程序，最终终于完成了这项任务。

最开始看到这个作业时，我就决心写一个手机上的 App。一方面是因为我最初想做一个平时可以用的 App，作为我和女朋友之间的聊天工具。另一方面是因为我想复习一下 Android 和 java 编程[5]。虽然本次实验中是在校园网环境下进行的，但之后我只要编写一个中央服务器程序，并简单修改一下本程序，平时就可以用我自己的 App 来聊天了。然而事实上，我其实是在给自己出一个很大的难题。开发的过程中我发现，要想把 Android 的界面做的精致，需要花很大的功夫，而我又十分追求完美，所以我不惜花很长时间来关注界面、功能、细节。最终算起来，这个项目花了我超过两周的时间。

虽然是这样，但是我却觉得很值得。本学期我共有 9 门大作业，我把每次大作业都当成一次锻炼自己编程能力的机会。我觉得学会一门技能的最好的办法就是用它完成一个项目，正如我选择 Android 和 java 编程，而不是停留在自己的舒适区，使用 C++ 或 C#。实验过程中，我不仅锻炼了 Android 编程能力，还学会了一些 PS 技术，最重要的是培养了解决问题的能力。发现问题时，我会积极查阅资料，寻找答案，最终总能够解决。

最后，感谢贾老师课上细心的讲解，感谢助教们在实验中的指导；感谢 Android 开发者编写详细的文档，感谢 StackOverflow 上网友们热心的解答！

[5] 曾经编写过简单的程序，但只是移植代码

参考文献

[1] “Room persistence library | android developers.” https://developer.android.com/topic/libraries/architecture/room. Accessed: 2019-1-5.

[2] “Bmob 后端云.” https://www.bmob.cn/. Accessed: 2019-1-5.

[3] “Android room with a view - java.” https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0. Accessed: 2019-1-5.

[4] “Adding swipe-to-refresh to your app | android developers.” https://developer.android.com/training/swipe/add-swipe-interface. Accessed: 2019-1-5.

[5] “I’m qq.” https://im.qq.com/download/. Accessed: 2019-1-5.

[6] “2,950,000+ free and premium vector icons. svg, png, ai, csh and png format..” https://www.iconfinder.com/. Accessed: 2019-1-5.

[7] https://github.com/HuTianQi/QQ. Accessed: 2019-1-5.

[8] “Mediarecorder | android developers.” https://developer.android.com/reference/android/media/MediaRecorder. Accessed: 2019-1-5.

[9] “Mediaplayer | android developers.” https://developer.android.com/reference/android/media/MediaPlayer. Accessed: 2019-1-5.

[10] https://github.com/koral--/android-gif-drawable. Accessed: 2019-1-5.

[11] “Take photos | android developers.” https://developer.android.com/training/camera/photobasics. Accessed: 2019-1-5.