计算机科学与技术学院

毕业设计

论文题目	快速文件传输 APP 的设计与实现——手机与手机互传		
学校导师	余颖	职　称	讲师
企业导师	罗超	职　称	
企业名称	华信教育科技有限公司		
学生姓名	张涛	学　号	20134350215
专　业	软件工程	班　级	本 13 软件 02 班
系 主 任	刘杰	院　长	刘振宇
起止时间	2016 年 6 月 5 日至 2017 年 5 月 22 日		

2017 年 5 月 22 日

目 录

快速文件传递APP的设计与实现——手机与手机互传

摘 要：往往会碰到人与人之间需要共享某份资料时，受当时环境的限制，比较难以传递，比如网络环境较差的环境下，手机无法联网，或者现有的方式不能满足当时的情况。本文针对这个问题，采用基于wifi的局域网通讯，设计了基于Android平台的文件传输并对历史文件进行管理的软件。

关 键 词： Android；文件传输；文件管理；SinVoice

Graphic Designer Design and Implementation of Express File Delivery APP-Delivery Between Mobiles

Abstract: People will face that they often need to share a certain information, subject to the constraints of the environment at that time, more difficult to pass, such as the network environment is poor environment, the phone can not be networked, or the existing way can not To meet the situation at that time. This paper for this problem, based on the wifi-based LAN communication, the design of the Android-based platform for file exchange and history file management software.

Key words: Android;File delivery;File management;SinVoice

引 言

移动互联网的时代已经到来，主要表现为成千上万的移动端应用被开发出来，人们可以在移动端方便地进行办公、学习和交流等，让人们之间的沟通变得更加便捷。随之而来往往会碰到人与人之间需要共享某份资料时，受当时环境的限制，比较难以传输，比如网络环境较差的环境下，手机无法联网，或者现有的方式不能满足当时的情况，比如虽然可以通过 QQ、微信等主流的社交软件可以传输资料，但临时性的接触不希望当时在 QQ、微信上互加好友等等。

目前世界上主流的移动开发平台有苹果公司旗下的 iOS 以及从发布以来便飞速发展的 Google 旗下的 Android。Android 是以 Linux 为基本核心，在其基础上进行精简并且拓展了 Linux 的基本功能，使其更加简单、易用，以适用移动平台便携、轻量的特点。Android 平台与 iOS 平台对比，其最主要的特点是开源的，这意味着只要获得商用的授权就可以免费用于商业化，而且还有完善的文件管理功能，其功能可以与 PC 相媲美。同时，Android 系统具备十分强大的功能扩展，还可以完整的与各大手机厂商应用商店的应用结合。因此，Android 从一发布到现在就深受手机用户的欢迎，根据分析机构 StatCounter 发布，Android 设备产生的流量在 2017 年 3 月首次超过了 Windows PC，可见 Android 系统占据了全球市场的主导地位。

本文的主要研究工作是，围绕 Socket 通信，基于 wifi 局域网，针对 Android 操作系统的核心技术，进行深入开展研究和分析，依据相关资料以及所学知识，提出快速文件传递 APP 的可行方案并实现。

第一章 绪论

1.1 研究背景和意义

文件传输是当下移动平台需求量极大的功能之一。智能手机的快速发展，使得人们步入了移动互联网的时代，极大地增加了人们相互沟通的频度以及便捷程度，而人们对于文件相互传输的需求也在这一过程中不断增长。往往会碰到人与人之间需要共享某份资料时，受当时环境的限制，比较难以传递，比如网络环境较差的环境下，手机无法联网，或者现有的方式不能满足当时的情况，比如虽然可以通过 QQ、微信等主流的社交软件进行文件传输，但临时性的接触，需要传输文件的双方又不希望在 QQ、微信上互加好友，或者文件较大没有 wifi，而流量资费又比较贵等等。

随着硬件性能以及软件的不断发展，文件的大小和手机的存储空间已经提升了很多。当下，一部 1080p 的电影会高达 1.6GB，而传统的传输方式，如蓝牙 4.0，理论上传输速度是 24Mbit/s，初步计算需要花费约 19 个小时才能传输完毕。因此，人们便有了快速进行文件传输的需求。尽管在 2016 年 6 月，蓝牙联盟发布了蓝牙 5，在有效传输距离上实现了较大的突破，理论上能达到 300 米，甚至还能结合 wifi 进行室内定位，但是这需要最新的硬件的支持，目前大多数设备使用的蓝牙版本仍然是 4.0 左右，普及还需时日。另外的，还有一些其他的方式，但是他们都或多或少存在一些缺陷，如 NFC 传输类型单一，需要硬件支持等。这些文件传输的方式大都不能满足人们如今的需求，因此，我们要寻找一个新的文件传输方式，使其能够实现快速稳定地传输大文件，并且能够支持传输多种文件的类型，以满足人们的需求。

WiFi Hotspot，即热点，是一个物理的地点，人们可以通过这个点连入外部网，是一种典型的使用 WiFi 技术，通过无线局域网来连入网络服务提供者的技术。虽然 WiFi 在传输过程中的通讯质量不是很好，安全方面不如蓝牙，传输质量也有待改进，但是 WiFi 相较于蓝牙的最大优势是，高传输速度，不需要繁琐的配对。据维基介绍，基于 IEEE802.11n 协议，传输速率可达 150Mbit/s。正是因为其传输距离远远大过蓝牙技术的范围，并且传输速度更快，操作简单，用户友好，支持多台设备同时连接等特性，成为传输方式的最佳选择。

因此，本文将利用 WiFi Hotspot 技术来完成端到端的文件传输，已解决无法使用社交软件或其他特殊情况下的文件传输问题。

1.2 国内外研究历史与现状

1.2.1 文件传输研究历史

蓝牙通信技术

在功能机时代，人们就有了文件传输的需求，当时最为实用、高效的技术，非蓝牙技术莫属。蓝牙技术最初的目的是实现一套无线通讯的解决方案，主要用于连接不同的设备平台，解决各个平台、不同设备之间的兼容性问题。

迄今为止，蓝牙已经发布至 5.0 版本，在经历了二十多年的版本的更新换代之后，蓝牙的传输速度在理论上已经从最初的 192Kbps 提升到了 24Mbps，大约提升了 128 倍，这进步是非常巨大的。同时，也考虑到了安全性，在通信时候，可以对连接的双方设置密码保护，通过这种方式来防止被其他方获取到。另外，兼容性也比较好，高版本的蓝牙能兼容低版本的蓝牙，可以连接多种不同的设备，只需要他们拥有相应的可以匹配的蓝牙模块即可，甚至可以通过蓝牙连接外设，如键盘、鼠标等。因此，蓝牙的主要优点在于兼容性强，安全性好。

尽管在传输速度上，蓝牙相对于最初版本已经提高了许多，然而，还是不尽如人意。在文件大小动辄几百兆的当代，显然不符合当今时代的用户需求。另一方面，蓝牙 5.0 版本传输速率得到了较大的提升，理论传输距离更是有了很大的突破，然而，最新的蓝牙技术需要硬件的更新换代，目前支持蓝牙 5.0 的智能手机几乎没有，未来第一批的支持者必然是旗舰级别的智能手机，普及还需要一段时日。因此，基于蓝牙技术的文件传输未被本论文列入研究范围。

其他通信技术

随后也出现了很多其他的文件传输技术，例如 NFC、红外线等等。由于红外线传输对硬件设备的要求较高，并且其传输条件比较苛刻（由于红外线的直射特性，不能被遮挡），很快就被其他的传输方式所取代。NFC 也是一种近距离的无线通讯方式。顾名思义，这种方式限制很大，只能在很近的范围内交换数据，实际上有效范围只

有 10 厘米。不仅如此，其传输速率也很慢，最快的一种大概每秒只能传输 50KB 的数据。因此，在实际生活中，NFC 常常用于传输体量比较小的内容，而若想要传输较大的文件，并不适合。

1.2.2 无线热点的应用

WiFi Hotspot，无线热点，是人们可以通过连接路由器的无线局域网而接入因特网的物理位置，Hotspot 与 WiFi AP 不同，WiFi AP 是一种提供无线网络的硬件设备，而 hotspot 往往是某个移动设备产生的，比如手机、平板等。此外，hotspot 可以与连接设备进行通信，而 WiFi AP 往往只是作为一个硬件接入点，类似于路由器的功能。公共接入无线局域网（LAN）首先由 HenrikSjödin 在 1993 年 8 月在旧金山的 Moscone 中心的 NetWorld + Interop 会议上提出。 Sjödin 并没有使用术语热点，而是使用可公开访问的无线局域网。

尝试创建公共局域网接入网络的第一个商业企业是在得克萨斯州理查森（Texas)理事会成立的公司，被称为 PLANCOM(公共局域网通信)。创业者 Mark Goode，Greg Jackson 和 Brett Stewart 于 1998 年解散了该公司，而 Goode 和 Jackson 则创建了 MobileStar Networks。该公司是第一个签署星巴克、美国航空公司和希尔顿酒店等公共接入地点。该公司于 2001 年被卖给德意志电讯公司，后者将该公司的名称改为“T-Mobile Hotspot”。那么那个时候，“热点”一词就进入流行的白话，作为一个可以公开访问的无线局域网可用的位置的参考。

热点常常被人们用来接入外部网络，比如自己的手机或者电脑无法通过数据流量上网，可以连接他人开启的热点进行上网，类似于连接无线路由器。然而它还有一个很棒的功能，就是用来进行局域网通讯。在这个热点创建的局域网中，所有连接的设备可以直接与连入这个热点的设备进行通信，其实就是局域网通讯，如果有配套的软件，有时候或许能解决许多意想不到的事情，比如，户外野炊，好友之间或许需要及时地分享照片，然而可能流量资费比较贵，多数人会选择回家后再分享。如果这时有个局域网通讯的 APP，通过这个 APP 就能够快速的分享文件，会增加不少乐趣，只是多数人没有注意到。

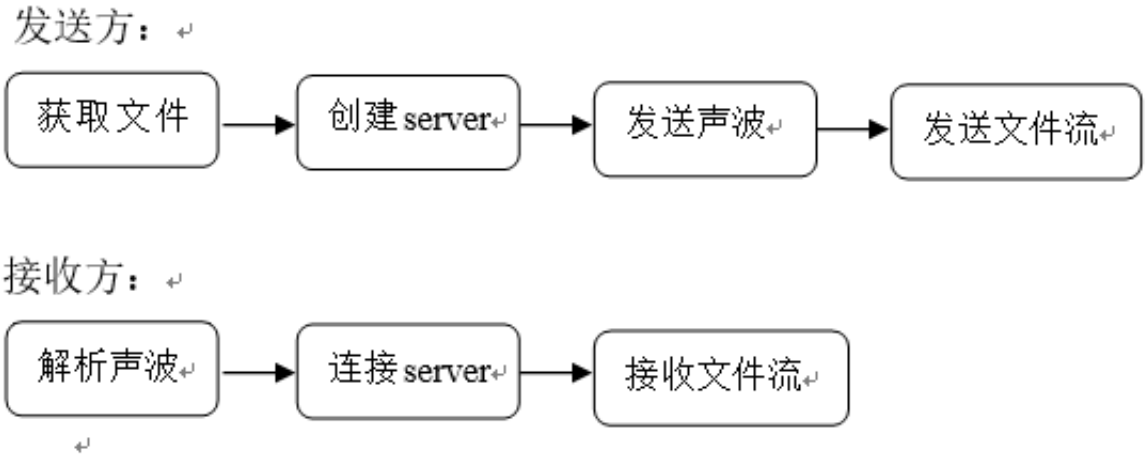
1.3 主要研究内容

本文致力于设计并实现一个快速的、稳定的、便捷的基于 C/S 结构的快速文件传递 APP，为尽量实现各模块的解耦，本 APP 采用 MVC 的架构进行实现。主要技术有 Socket 通信、声波传输等。在准备了大量的文件传输、Socket 编程和声波传输的相关理论和核心技术的前提下，着重研究了局域网的环境下，进行端到端的文件传输。总体来说，基于 C/S 结构的快速文件传递 APP 的传输流程图如图（图 1.1）所示，核心研究内容如下：

1）对第三方的声波通信 lib——SinVoice 进行研究和分析，运用它提供的 API 传输 server 端的 ip 地址到 client 端，从而建立基于 C/S 模式的 socket 连接，为文件传输模块做准备；

2）在近场、不通过外网的情况下，利用 WiFi Hotspot 技术，通过 C/S 模式的 socket 通信，进行文件传输；

3）利用自己封装的文件操作的工具类，提供对历史获取的文件进行删改、再次传输等操作；



1.1 文件传输关键流程

1.5 论文的结构安排

论文共分 章撰写，具体组织结构如下：

第一章：绪论。介绍了论文的选题背景，阐述了文件传输技术的研究历史和现状，以及本文的选题意义。

第二章：专业知识和技术介绍。主要对 APP 中设计和开发中所用到的相关的专业知识和技术做了详细的介绍，并对其原理进行了详细的说明。。

第三章：系统需求分析。根据快速文件传递 APP 的基本要求，对其在功能上与非功能上进行详细地需求分析。在准确的了解和定位我们研究快速文件传递 APP 所需要的技术路线之后，做出系统的、符合实际的、完善的需求分析，同时，在开发的过程中，为了方便后续的优化和维护，需要采用 MVC 架构进行 APP 的开发。

第四章：系统设计。根据系统需求分析，对快速文件传递 APP 进行概要设计和详细设计。先列举出快速文件传递 APP 的总的结构、功能，然后列举出快速文件传递 APP 实现所需要的子系统、子模块，以及完成这些模块所需要的功能和非功能需求的清单列表，基于这个需求清单做出进一步的完善分析与设计。

第五章：系统实现。根据系统需求分析和系统设计，对快速文件传递 APP 的各个模块进行实现。

第六章：总结与展望。根据写作论文以及实现 APP 的基本情况，结合本设计的工作结果进行总结，并指出通过对本 APP 的设计与开发的收获，以及本 APP 存在的一些问题，最后是对文件传输技术未来的应用前景进行展望。

第二章 专业知识与技术介绍

本章主要对本论文中用到的一些相关技术的进行简要介绍，为后续步骤做好前期的技术准备。

2.1 Android

由于手机市场前景广阔，并且发展迅猛，不同厂商的定位跟出发点会有不同，使得手机操作系统也出现了不同的种类。从市场占有率来看，根据分析机构 StatCounter 发布，Android 设备产生的流量在 2017 年 3 月首次超过了 Windows PC，这标志着 Android 不仅是移动操作系统的霸主，还成为了全球第一大操作系统。因此，本文在手机系统的选择上，以 Android 为平台，进行软件的设计以及开发。

2.1.1 Android 简介

Android，中文名称安卓，是一个基于 Linux 内核的开放源代码的移动端操作系统。它是由 Google 成立的 Open Handset Alliance（OHA，开放手持设备联盟）持续领导与开发，主要设计用于触屏移动设备，如智能手机和平板电脑与其他便携式设备。

最初，Android 是由 Andy Rubin 等人开发制作，最初开发这个系统的目的是创建一个数码相机的先进操作系统；但是后来发现市场需求不够大，加上智能手机市场快速成长，于是，Android 被改造为一款面向智能手机的操作系统。于 2005 年 7 月 11 日，被 Google 公司收购。2007 年 11 月，Google 与多家硬件制造商、软件开发商及电信运营商成立开放手持设备联盟来共同研发改良 Android，随后，Google 以 Apache 免费开放源代码许可证的授权方式，发布了 Android 的源代码，开放源代码加速了 Android 普及。

2.1.2 Android 架构

在 APP 的实现过程中，会调用到 Android 系统不同层次的资源甚至于硬件资源，因此在实现本 APP 前，有必要了解 Android 的架构。Android 系统是基于 Linux 内核的，它的体系结构如图 2.1 所示：

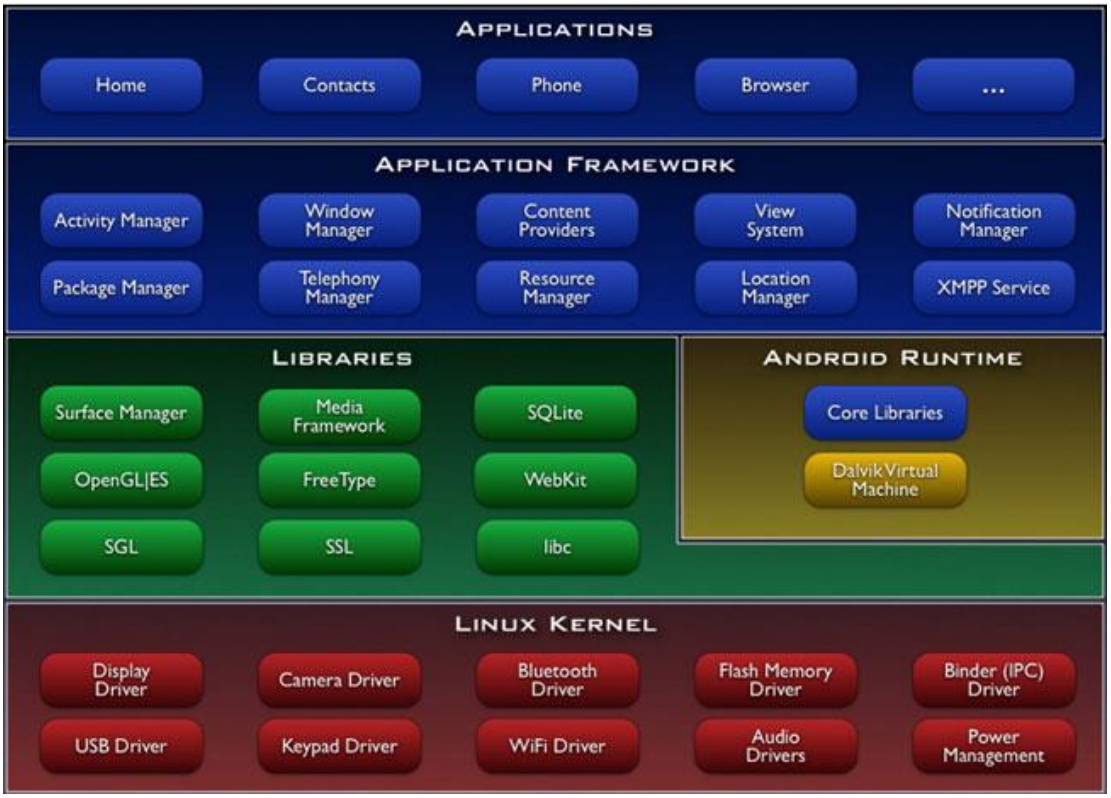


图 2.1 Android 系统结构图

第一层：图中最下面的一层，即最底层，这一层是 Linux 内核层，我们知道安卓是基于 linux 的，因此，这层是安卓的最底层。这一层提供了很多的系统服务，如进程管理，内存管理，设备管理（如摄像头）等。对于应用程序开发人员，该层为软件与硬件之间增加了一层抽象层，对程序员来说是透明的，使得在开发人员在开发过程中不需要考虑底层硬件的细节。

第二层：这一层分为系统函数库和安卓运行时两个部分。为应用程序开发人员提供了许多使用 C/C++编写的函数库，供他们使用，方便软件的开发；安卓运行时则提供了运行安卓程序的虚拟机，它与运行 Java 程序的基于栈存储的 jvm 不同的是，dalvik 是基于寄存器的，运行速度很快，并且每个程序在运行时会分配一个虚拟机，这意味着他们可以高效地运行且互不影响。比如某个程序在运行时奔溃了，但是另

外的程序仍可以正常运行。

第三层：这一层对第二层进行了封装，将一些系统功能封装成可以重用的组件，并提供想应的调用方法，如 ActivityManager 等，大大降低了开发程序的难度。

第四层：这一层是安卓系统的最上层，也就是用户能看到的，主要指各种应用程序，程序开发人员就在这一层使用 Java 语言进行开发。包括一些系统自带的应用程序，如文件管理、短信等。

在实现 C/S 连接时，需要有 WiFiManager 来对 WiFi 模块的资源进行管理，这就需要调用最底层的 WiFi Driver 来实现对 WiFi 硬件的管理，在获取并设置 WiFi 的配置信息时，需要通过 Java reflect 来实现；在获取本地文件列表时，由于 Android 的文件信息会保存到 sqlite 数据库，利用这一特性，可以通过查询语句实现对本地文件的快速获取，这时可以使用位于第三层的内容提供者对位于第二层的本地数据库进行读取来获取文件信息；在实现 Android 界面时，可以使用 Activity Manager 与 Surface Manager 来实现传输过程的动画效果等。

2.2 SinVoice

SinVoice 是一个提供声波编解码的第三方库，为多个主流平台提供接入。目前已经有多个程序使用到了这项技术。主要是通过声音信号对数据进行编码，而这个声音信号的频率是单一的、固定的。假定某一个频率对应一个数字，比如用 100Mhz 表示数字 1。在传递内容前，先对目标内容进行编码，即将其像密码一样翻译成对应的生波段。然后，通过麦克风或扬声器来播放这一生波段。接收方则需要开启录音，在录音完成后，也像密码一样根据预定义的规则对其解密，这里就根据声音的频率解码出对应的数据。而 SinVoice 就是完成对内容与声音信号进行加密解密转化的一个函数库。

2.3 MVC 架构

MVC 模式（Model-View-Controller）是软件工程中的一种软件架构模式，把软件系统分为三个基本部分：模型（Model）、视图（View）和控制器（Controller）。MVC 模式的目的是实现一种动态的程序设计，是后续对程序的修改和扩展简化，并且

使查询某一部分的重复利用成为可能。除此之外，此模式通过对复杂度的简化，使得查询结构更加直观，便于开发人员阅读。便于UI界面部分的显示和业务逻辑，数据处理分开。MVC针对Android端的职责如下：

M（模型）层：主要做一些业务逻辑的处理，比如数据库的存取操作，网络操作，复杂的算法，耗时的任务等；

V（视图）层：主要包括xml布局，Activity中跟UI显示相关的部分，处理并显示Model层中的数据结果。

C（控制）层：主要包括人机交互的处理，而 Activity 是处理这些的，因此可以认为Activity是控制器，处理用户输入，并向Model层发送数据请求等。

MVC框架降低了耦合性。利用MVC框架是的视图层（V）和模型层（M）可以很好地分离，这样就达到了解耦的目的，减少模块之间代码的相互影响。

调高了可扩展性。当 MVC 中某个部分需要改动时，其他的部分不需要修改，当天件新的需求时，代码可扩展就可以大大减少代码的修改率，降低了其他模块重复开发的成本。

职责划分明确。划分为三个主要模块，有利于代码的阅读，便于开发人员理清开发思路，从而方便开发人员进行开发。

第三章 系统需求分析

本章从快速文件传递 APP 的功能性需求和非功能性需求出发，对整个系统的需求进行可行性分析，形成明确的需求规格说明，为系统设计做准备。

3.1 任务概述

在过去的一段时间里，随着科技的进步与人们生活水平的的提高，智能手机得到快速的普及，人们使用更多的设备已是手机。因此，人们经常需要通过手机共享某份资料，有时受当时环境的限制，或许会比较难以传递。比如网络环境较差的环境下，手机无法联接外网，或者现有的方式不能满足当时的情况，比如虽然可以通过 QQ、微信等主流的社交软件可以传递资料，但临时性的接触不希望当时在 QQ、微信上互加好友；或者文件较大，又恰好没有 wifi，而流量资费比较贵等等。

本项目旨在利用基于 WiFi Hotspot 的局域网进行文件的传输，以支持对不同场景下，无论是否可以连接外部网络，用户都可以自由地分享文件给身边的人。同时，用户可以对历史接收的文件进行管理，方便用户对历史接收文件，快捷地进行管理、再次分享等操作。

1） 实现文件传输功能；

2） 实现文件管理功能；

3） 简化交互逻辑；

3.1.1 用户特点

本 APP 的受众范围比较广泛，主要针对用户无需使用因特网即可快速、稳定、高效地互相传递文件。比如，工作中，同事之间需要互传资料时，可能会遇到由于员工数量多，无法连接 WiFi，而文件又比较大，使用传统的蓝牙传输，不仅传的比较慢，可能匹配的步骤也比较繁琐；生活中，或许会遇到不愿意透露社交账号，但是又急需互传文件等资料时，一时间苦于找不到合适的方式来解决等等情况。通过这款 APP，我们可以很方便。快捷、稳定地传输文件，也无需考虑泄漏个人的信息

等风险。

3.2 需求规定

3.2.1 系统功能结构图

图 3.1 系统功能框图

3.2.2 用例图

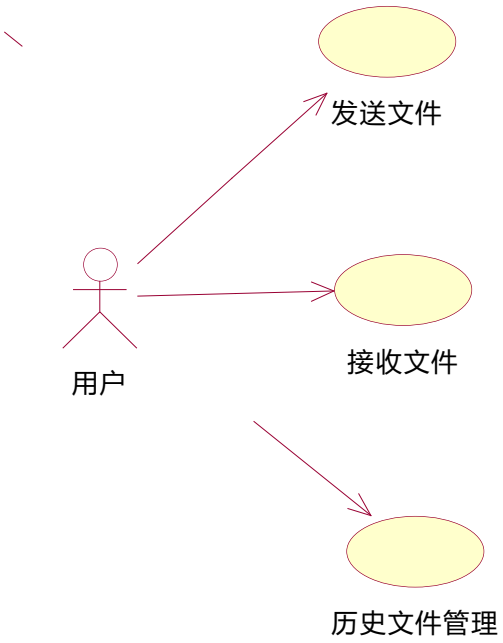


图 3.2 系统用例图

3.2.3 用例描述

表 3.1 用例分析

用例编号	用例名	用例描述	主要参与者
SRS_case1	发送文件	用户点击发送按钮后，进入本地文件列表，选择完目标文件后点击完成即可进行发送	用户
SRS_case2	接收文件	用户点击接收按钮后，进入发送页面，后台自动完成与发送方的连接并获取文件	
SRS_case3	历史文件管理	用户可以在 APP 中，对历史接收到的文件进行删改等操作，还可以进行再次分享操作	

3.3 详细功能需求

快速文件传递 APP 主要实现了发送模块、接收模块以及历史文件管理模块。要求能够快速、稳定、方便地实现文件的互传，并且能够方便地对历史接收文件进行管理。不仅仅在文件传输上要求高效，还需要在人机交互的逻辑上，优化交互逻辑，从而提高用户的体验。

3.3.1 发送文件（SRS_case1）功能需求分析

表 3.2 发送文件功能清单

子功能	子功能描述	所属用例

点击发送按钮	跳转到文件列表页面	发送文件
显示本地文件列表	在文件列表页面，读取本地文件并进行，方便用户快速选择	
选择文件	用户可以选择发送一个或多个文件	
发送文件	用户选择完文件后，无需再操作，程序会在后台自动完成发送操作。先创建 server，再通过声波发送 ip，接收方接收到之后，会自动连接，连接成功时候便会传输文件。	

用户在文件列表页面完成目标文件的选择之后，可以点击完成进行文件的传递。若遇到发送完成或者发送失败并且超时，则会结束发送这一流程，否则重新发送。这个用例的人机交互流程图如下：

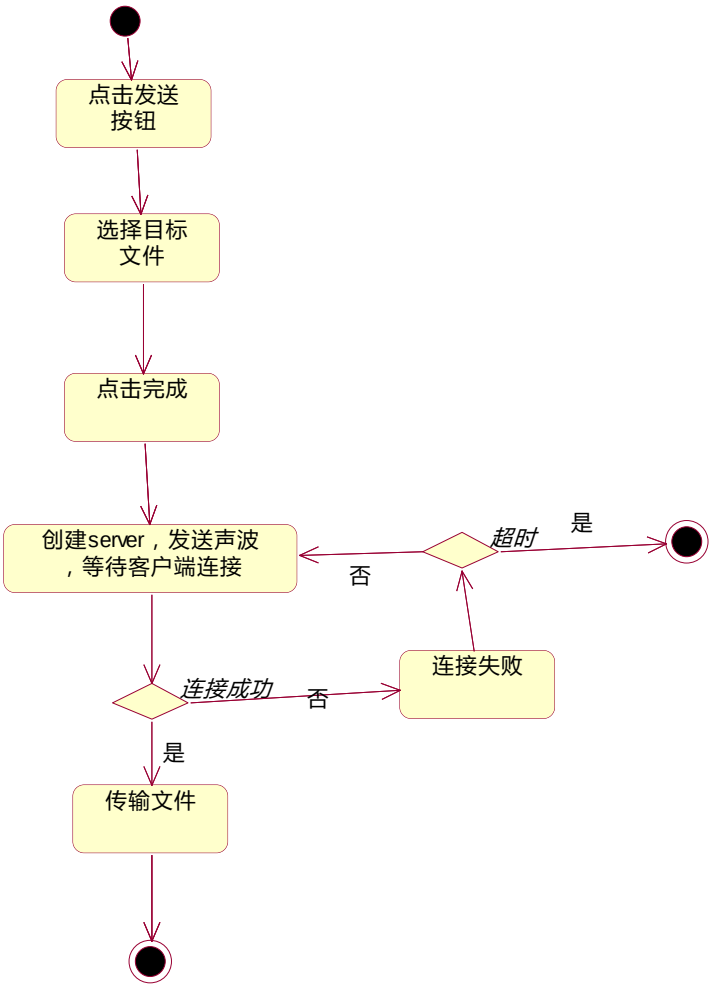


图 3.3 发送文件流程图

3.3.2 接收文件（SRS_case2）功能需求分析

表 3.3 接收文件功能清单

子功能	子功能描述	所属用例
接收	跳转到接收页面	手机与手机传输
解析	解析声波，获取连接信息	
连接	根据连接信息连接服务器	
分类	接收方获取到文件后，对文件进行分类存储	

用户选择接收后，程序会完成连接发送端并完成文件传输操作，无需用户再操作，这一过程的活动图如下：

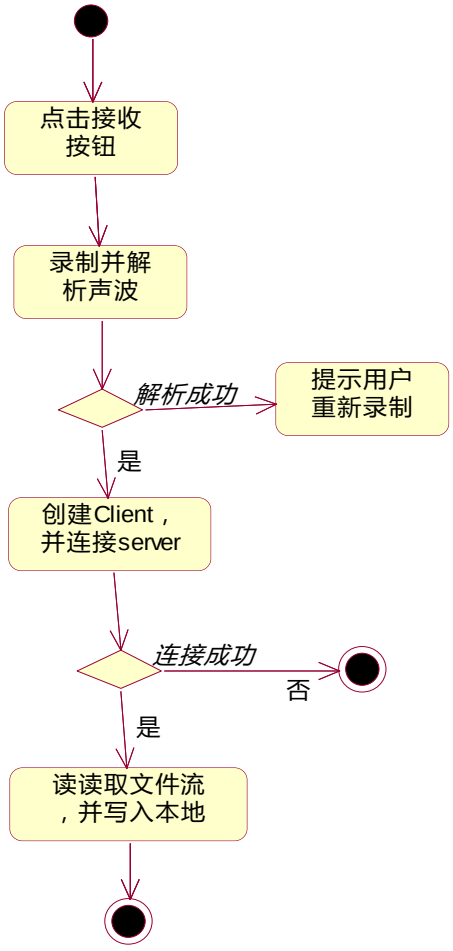


图 3.4 接收文件活动图

3.3.3 历史文件管理（SRS_case3）功能需求分析

为了方便用户对历史接收文件进行管理、再次分享等，本模块将用户历史接收的文件，进行了获取并展示，提供了再次分享等功能。

表 3.4 文件管理清单

子功能	子功能描述	所属用例
文件复制	能够复制目标文件	手机端文件管理功能
剪切	能够剪切目标文件	
粘贴	能够粘贴已经复制或者剪切的目标文件	
重命名	能对目标文件进行重命名	
复制路径	能复制文件的路径，方便用户去系统文件管理器查找	
无网发送	能将用户选中的历史文件再次通过本 APP 发送给他人	
有网发送	能将用户选中的历史文件通过因特网发送给 QQ 或微信等好友	
文件详情	能获取文件的详细信息，并展示给用户	
侧滑删除	左滑文件，显示垃圾桶图标，点击该图标完成删除	

由于操作流程类似，这里以文件复制为例，用户长按目标文件，选择复制，然后进入目标文件夹进行粘贴，活动图如下：

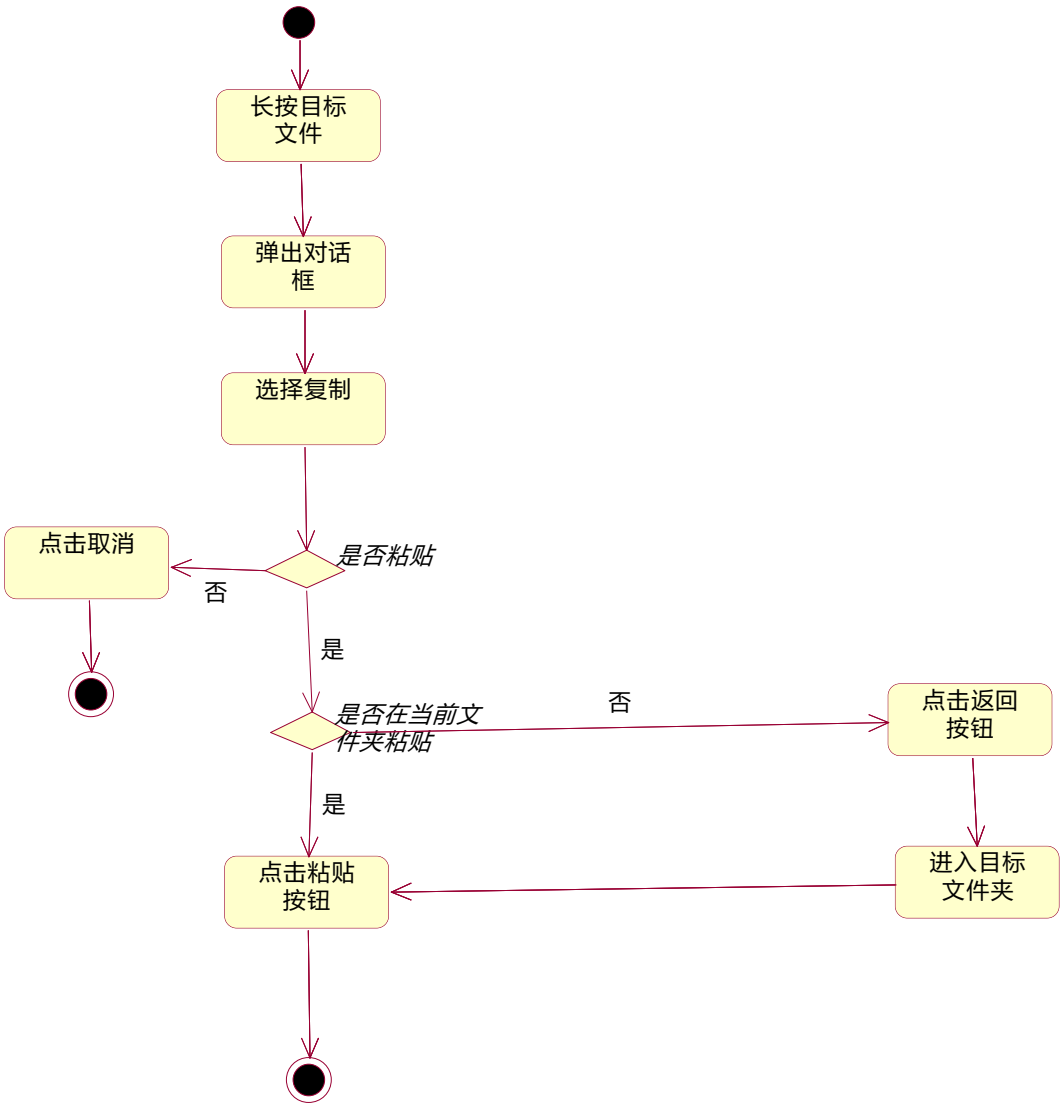

长按目标
文件
弹出对话
框
选择复制
是否粘贴
点击取消
否
是
是否在当前文
件夹粘贴
否
点击返回
按钮
是
点击粘贴
按钮
进入目标
文件夹

图 3.5 文件管理流程图

侧滑删除文件，用户左滑目标文件，点击删除文件的图标，活动图如下：

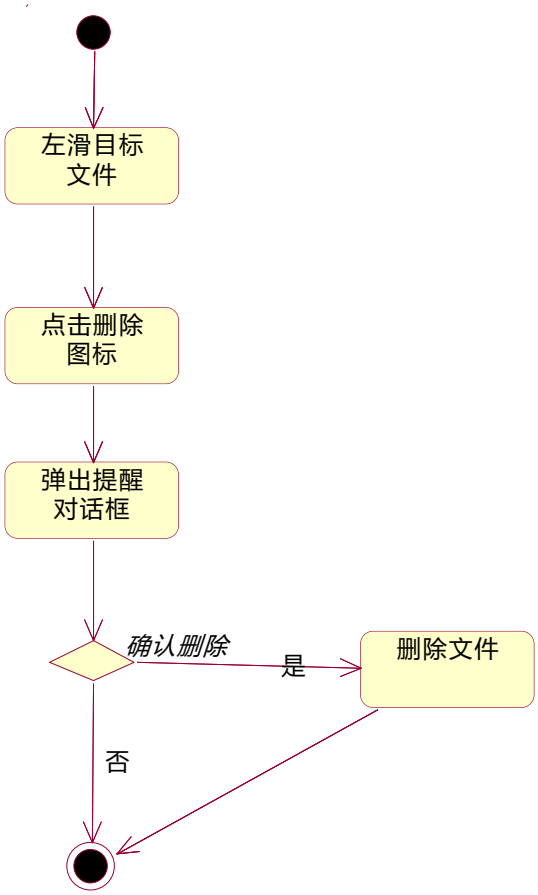


图 3.6 “删除文件“活动图

第四章 系统设计

本章主要确定具体的 APP 体系结构，系统的架构模式以及设计系统各个模块的实现。

4.1 系统体系结构设计

依据需求分析，快速文件传递 APP，是基于 WiFi 局域网通信，是通过近距离的无线射频信号进行信息传递的方式。因此，使用本 APP 时要在相近的地点。按照上一章的需求分析，可设计如下系统结构：

整个系统模块如下图：

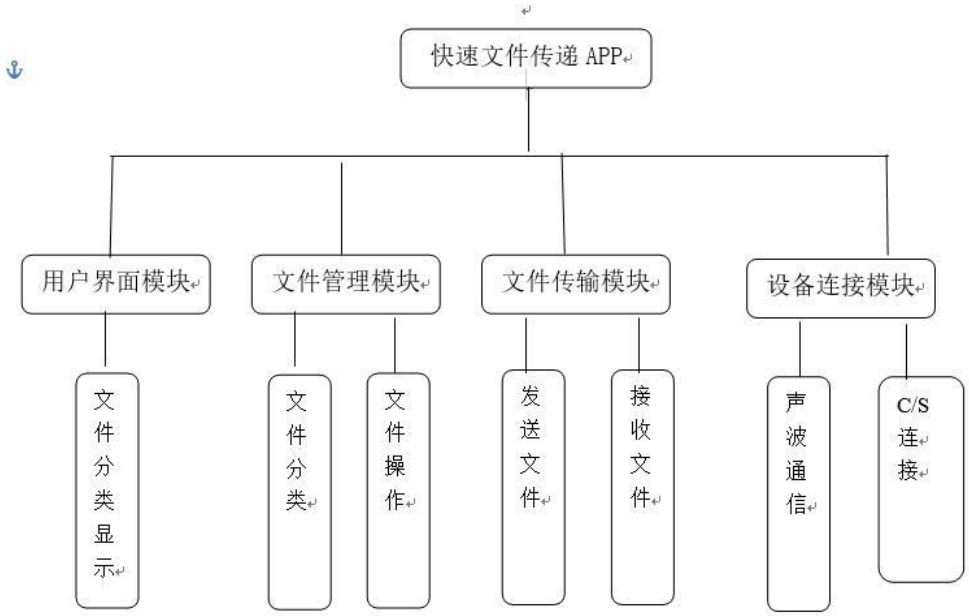


图 4.1 系统模块图

4.1.1 系统结构

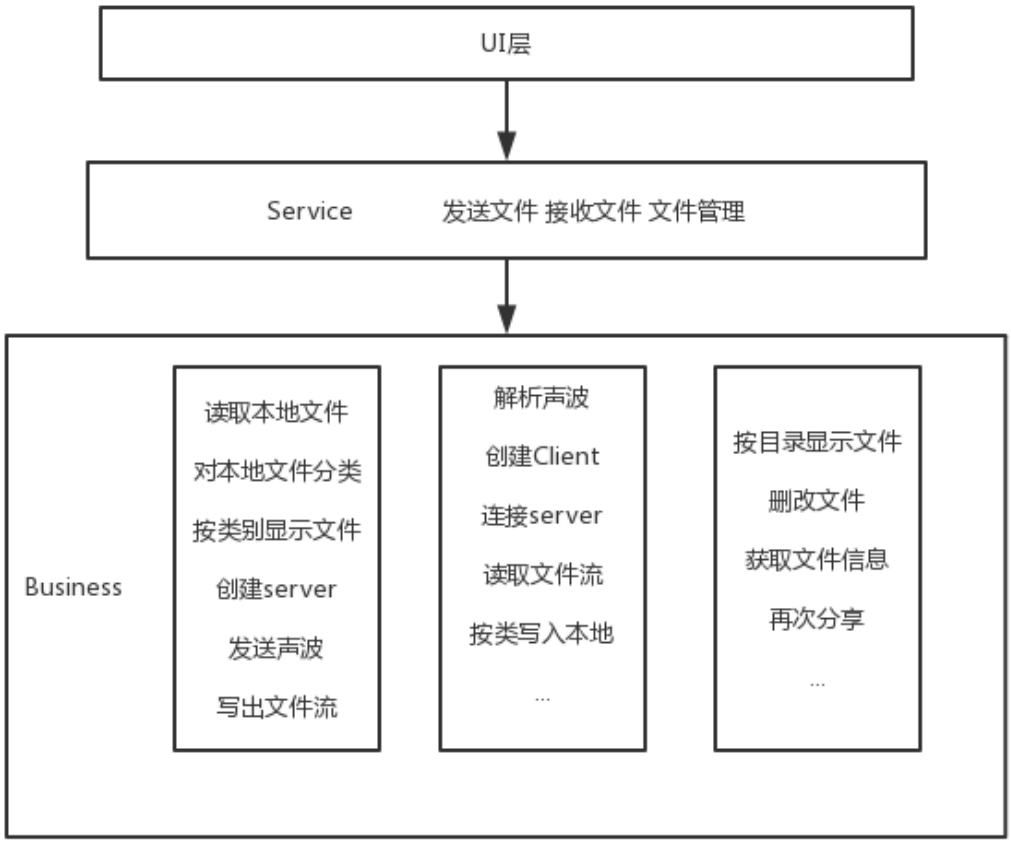


图 4.2 系统结构图

4.1.2 设备之间的传输体系

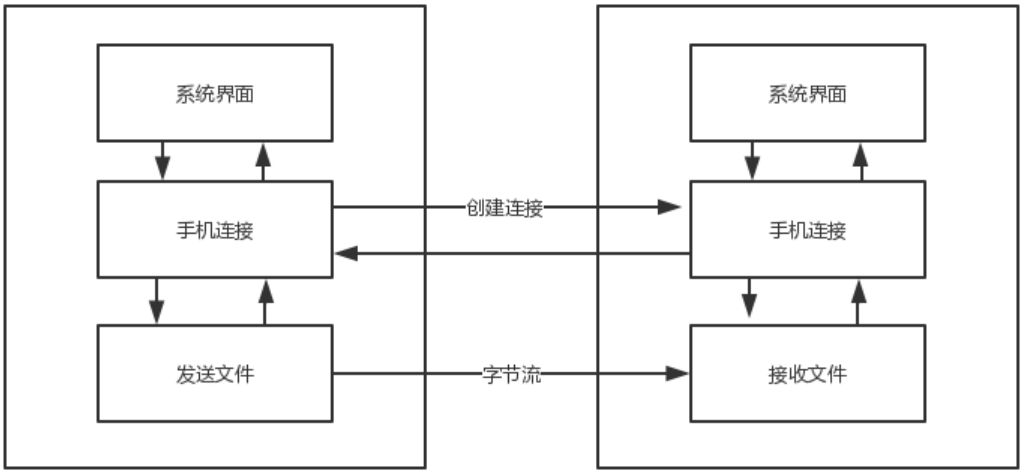


图 4.3 设备间传输文件结构图

4.2 各模块功能设计

4.2.1 设备连接模块

连接分为两个部分，一个是发送端通过声波发送自己的 ip，然后接收端录音并解析 ip；另一个是 C/S 连接模块，接收端获取到发送端的 ip 后，通过 ip 和事先规定的端口号进行连接的一个模块。

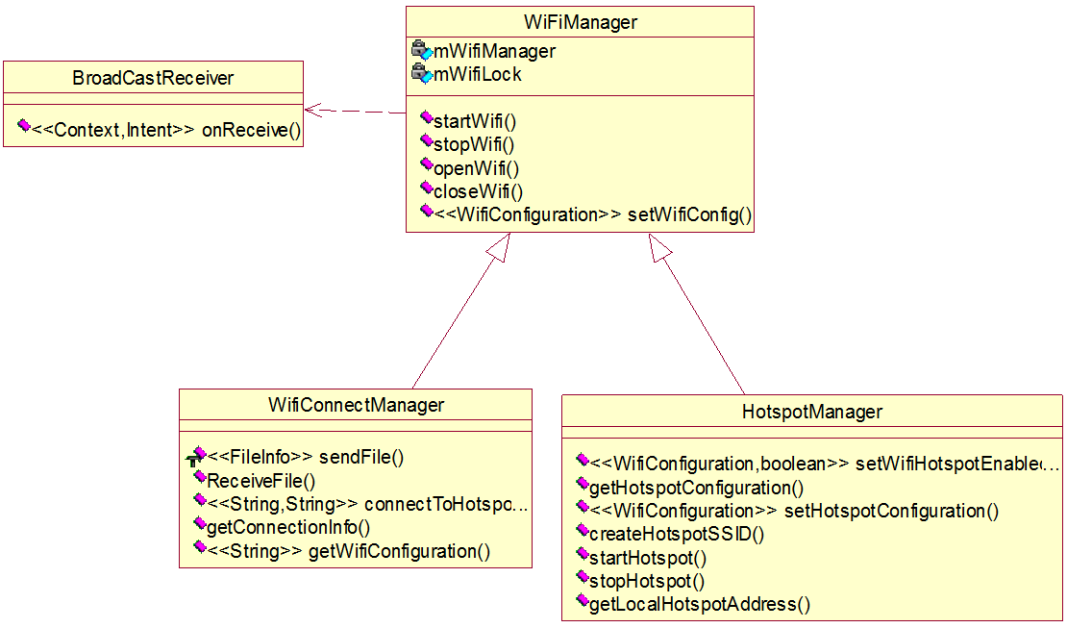


图 4.4 连接模块类图

这一模块的设计思路是，发送方先获取自己的相关信息，然后根据协议对内容进行封装，之后通过声波发送这段协议；接收方则开启录音，利用 SinVoice 提供的 API 解析这段声波，通过监听回调获取解析结果。最后依据协议，解析声波内容，获取 server 的 ip。协议内容，wifi 名字:ip

4.2.2 传输模块

传输模块主要负责文件流的读取、传输等，设计如下：

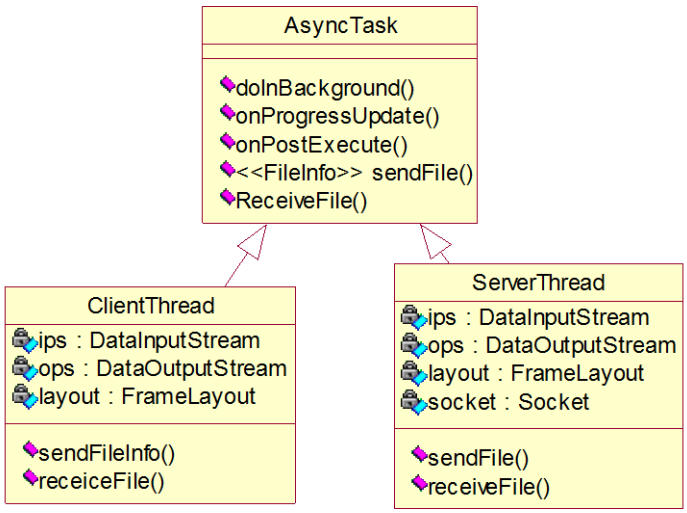


图 4.5 服务器与客户端类图

4.2.3 文件管理模块

这个模块提供了对用户历史接收的文件进行操作的功能，包括文件的删改、再次分享等一系列文件操作。

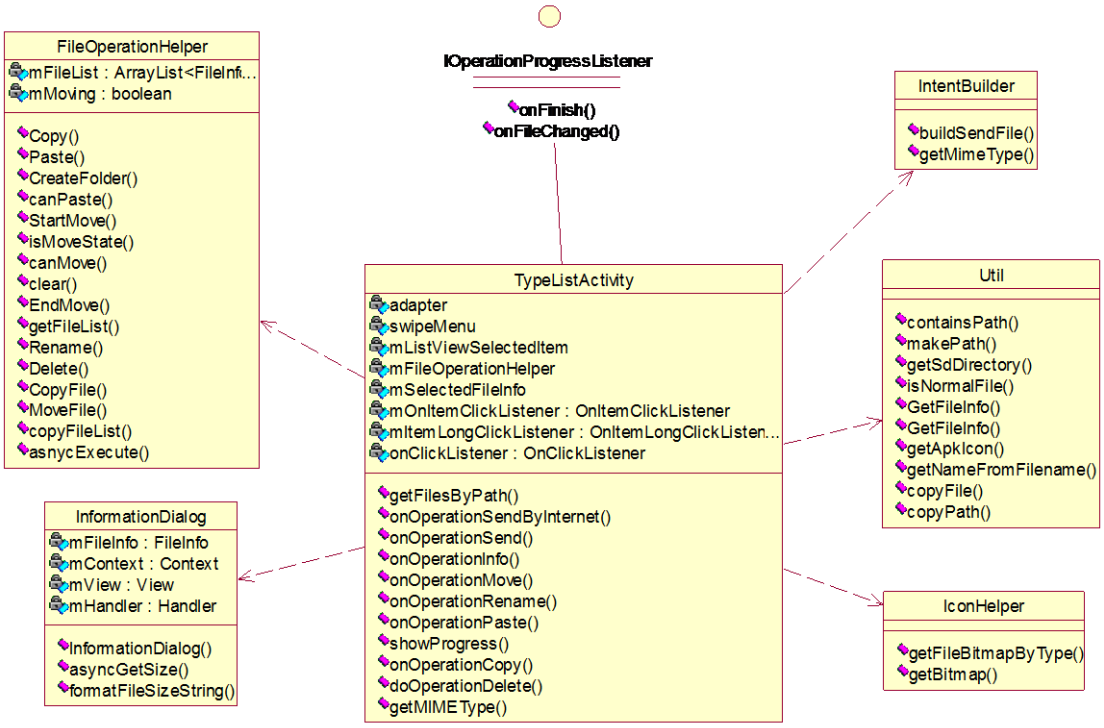


图 4.6 文件管理类图

定义了文件操作帮助类，使得对与 UI 无关的文件的操作在逻辑上得到了抽象和统一。

4.3 可靠性设计

在可靠性方面，本 APP 基于 Socket 通信，将目标文件转换成流在网络中进行传输，而这种通信方式则是对 TCP/IP 协议的封装和应用。通过这个协议，可以对文件进行可靠的在网络中进行传输。

4.4 用户界面设计

1. 文件传输页面，需要提供发送、接收按钮。在本 APP 中，无论是发送方，还

是接收方，操作都很少，减少用户的学习成本。发送方，只需点击发送，继而选择目标文件，最后点击完成即可完成发送操作；接收方则更加简单，只需点击接收，然后依据权限提示，打开录音功能即可。剩下的，连接收发双发的操作，文件的写入目录等，均由程序自动完成，无需用户任何操作。同时，本 APP 还设计有传输过程的动画，优雅舒适，增加用户使用的愉悦感。

2.历史文件管理页面，为了快速、方便、明了的对历史文件进行管理，需要对文件进行分类。同时，为了减少用户学习的成本，需要对文件操作的交互进行简化，为此，我们设计了对目标文件长按，即可唤出文件操作列表，方便，易用。对文件的删除，我们加入了侧滑删除的模块，操作简便、明了，延续用户的使用习惯，增进人机交互体验。

4.5 故障处理说明

出错名称	系统输出信息	处理方法
文件操作错误	“不能复制文件夹，请重新选择！”	保持在文件选择页面
连接失败	“连接超时，请重新连接！”	进入等待连接页面

第五章 系统实现

5.1 连接模块

5.1.1 声波传输

这一模块，主要利用了第三方的库——SinVoice 来实现，根据提供的 API，可以完成如下发送声波的部分：

发送声波

先获取本地 ip，然后通过声波发送。

```
public String getLocalIpAddress() {
    WifiManager wifiManager = (WifiManager) getSystemService(WIFI_SERVICE);
    WifiInfo wifiInfo = wifiManager.getConnectionInfo();
    // 获取32位整型IP地址
    int ipAddress = wifiInfo.getIpAddress();

    // 返回整型地址转换成“*.*.*.*”地址
    return String.format("%d.%d.%d.%d", (ipAddress & 0xff),
            (ipAddress >> 8 & 0xff), (ipAddress >> 16 & 0xff),
            (ipAddress >> 24 & 0xff));
}
```

图 5.1 获取 ip

```
    private void sendAddress() {
        try {
            String localIpAddress = getLocalIpAddress();
            Log.e("本服务器的ip", localIpAddress);
            StringBuilder ssid = new StringBuilder();

            ssid.append("whqtest").append(":"); // 文件头信息
            // 加上文件头信息
            ssid.append(localIpAddress);
            Toast.makeText(App.context, localIpAddress, Toast.LENGTH_SHORT)
                    .show();
            byte[] strs = ssid.toString().getBytes();

            if (null != strs) {
                int len = strs.length;
                int[] tokens = new int[len];
                int maxEncoderIndex = mSinVoicePlayer.getMaxEncoderIndex();
                LogHelper.d(TAG, "maxEncoderIndex:" + maxEncoderIndex);
                String encoderText = ssid.toString();
                for (int i = 0; i < len; ++i) {
                    if (maxEncoderIndex < 255) {
                        tokens[i] = Common.DEFAULT_CODE_BOOK
                                .indexOf(encoderText.charAt(i));
                    } else {
                        tokens[i] = strs[i];
                    }
                }
                mSinVoicePlayer.play(tokens, len, false, 2000);
            } else {
                mSinVoicePlayer.play(TOKENS, TOKEN_LEN, false, 2000);
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

图 5.2 发送声波

为了使连接稳定，在进入 onCreate（）方法之后，即立即先关闭 wifi，启动 wifi hotspot，然后启服务器。在 onResume（）方法中，开始循环等待 wifi hotspot 启动，为了避免死循环破坏用户体验，每次循环都失眠一百毫秒，然后循环超过 10 次，就退出循环。若启动成功就执行发送声波的代码，把服务器 IP 通过声波发送出去。

```
int time = 0;
while (!accessPoint.isWifiApEnabled()) {// 如果WiFi没有被允许使用，就进入循环，等待用户允许；在规定时间内，被允许后，则跳出循环
    // 堵塞
    try {
        time++;
        if (time > 10)
            return;
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
// 跳出循环，即用户允许了
sendAddress(); // 启动完成后，把本机ip和服务器端口号用声波发送
```

图 5.3 超时不发送

解析声波

先接收声波，然后解析声波

```
/**
 * 接收数据的方法
 *
 * @return
 */
private String receiveVoice() {
    mRecognition.start(TOKEN_LEN, mIsReadFromFile);
    return null;
}
```

图 5.4 开始接收声波

```
@Override
public void onSinVoiceRecognition(char ch) {
    mHanlder.sendMessage(mHanlder.obtainMessage(MSG_SET_RECG_TEXT, ch, 0));
}
```

图 5.5 解析声波

将解析结果存储，方便后续使用

```
case MSG_SET_RECG_TEXT:
    char ch = (char) msg.arg1;
    // mTextBuilder.append(ch);
    activity.mRecgs[activity.mRecgCount++] = ch;
    break;
```

图 5.6 存储每次的解析结果

最后，按照既定的协议解析声波，获取 ip 这一目标信息

```
case MSG_RECG_END:
    LogHelper.d(TAG, "recognition end gIsError:" + msg.arg1);
    if (activity.mRecgCount > 0) {
        byte[] strs = new byte[activity.mRecgCount];
        for (int i = 0; i < activity.mRecgCount; ++i) {
            strs[i] = (byte) activity.mRecgs[i];
        }
        try {
            String strReg = new String(strs, "UTF8");
            Toast.makeText(App.context, strReg, Toast.LENGTH_LONG)
                    .show();
            if (null != mAct) {
                // 得到ip后，链接对方的服务器
                Log.e("接收到的IP: ", strReg);
                // 校验
                String[] code = strReg.split(":"); // 文件头信息
                String ssid = null;
                String ip = null;

                if (code.length > 1) {
                    ssid = code[0]; // ssid
                    ip = code[1]; // 正文
                }
                // 链接
                activity.wifiConnect.connectToAccessPoint(ssid,
                        null);

                while (!activity.mWifiManager.getConnectionInfo()
                        .getSSID().contains(ssid)) {
                    Thread.sleep(50);
                    // 等待链接好ap
                }
                activity.mHanlder.sendEmptyMessage(200);
                // 启动asyncTask
                // ap的地址是写死的,就是下面这个地址
                new ClientThread(frame).execute("192.168.43.1");
            }

        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    break;
```

图 5.7 依据协议解析声波内容并获取 ip

5.1.2 C/S 连接

为了确保能够连接到服务器的热点，不管客户端有没有启动 wifi hotspot，都先做个判断，把 wifi hotspot 关闭，然后打开 wifi，在 onResume（）中开启线程调用接收声波的代码，接受服务器发来的信息，然后通过 Handler 把消息加入到 MessageQueue，转到主线程 UI，然后开启线程连接服务器的 wifi hotspot，来判断

用户是否打开热点，并给出相应提示。

发送端（即服务端）：

先关闭 WiFi 连接

```
accessPoint = new AccessPointManager(this);
// 关wifi，开热点
accessPoint.stopWifi();
if (!accessPoint.isWifiApEnabled()) {
    accessPoint.startWifiAp();
}
```

5.8 关闭 WiFi

然后启动热点，需要先按照预先设定的配置来配置 WiFi 热点。如 WiFi 名称，是否设置密码等。

```
/** create WiFi hot spot */
public boolean startWifiAp() {
    // close WiFi
    // if(mWifiManager.isWifiEnabled()){
    closeWifi();
    // close the open WiFi hot spot
    if (DeviceUtils.isHTC()) { // 2013-12-31 add special for htc
        enableHtcHotspot(getWifiApConfiguration(), false);
        return setHtcHotspot();
    }

    if (isWifiApEnabled())
        setWifiApEnabled(getWifiApConfiguration(), false);
    // envoke the hot spot need to be created
    acquireWifiLock();
    WifiConfiguration wifiConfig = new WifiConfiguration();
    // 手机型号
    // String model = android.os.Build.MODEL + "_ap";
    createWifiApSSID("whqtest"); // ap名称
    wifiConfig.SSID = getWifiApSSID();

    // wifiConfig.preSharedKey = DEFAULT_PASSWORD;
    // setWifiConfigAsWPA(wifiConfig); // have password
    setWifiConfig(wifiConfig); // no password
    return setWifiApEnabled(wifiConfig, true);
}
```

图 5.9 根据配置创建 WiFi hotspot

最后创建基于此 WiFi 热点的 serversocket，等待客户端连接

```
try {
    serverSocket = new ServerSocket(Constant.PORT);
    // String hostName = InetAddress.getLocalHost().getHostName();
    Log.e("tag", "服务器启动成功");
    handler.post(new Runnable() {

        @Override
        public void run() {
            Toast.makeText(App.context, "服务器启动成功", Toast.LENGTH_SHORT)
                    .show();
        }
    });
```

图 5.10 创建 server

接收端（即客户端）：

先按照预先设定的配置连接 WiFi。如果没有找到目标 WiFi，则设定重连时间，超时则放弃连接，结束接收

```
    // 链接
    activity.wifiConnect.connectToAccessPoint(ssid,
            null);

    while (!activity.mWifiManager.getConnectionInfo()
            .getSSID().contains(ssid)) {
        Thread.sleep(50);
        // 等待链接好ap
    }
    activity.mHanlder.sendEmptyMessage(200);
    // 启动asyncTask
    // ap的地址是写死的,就是下面这个地址
    new ClientThread(frame).execute("192.168.43.1");
}
```

图 5.11 根据声波模块解析出的 WiFi 的 ssid 连接目标 WiFi

连接目标 WiFi 成功后，解析声波中的 ip 地址，连接 server

```
        InetSocketAddress address = new InetSocketAddress(value[0],
                Constant.PORT);
        Connect con = Connect.getInstance();

        // con.disConnect();
        int times = 0;
        while (!con.connectServer(address, Constant.PORT)) {
            Thread.sleep(100);
            times++;
            if (times > 100) {
                // 协议：第一位 是进度百分比，第二位是文件名，第三位是信号
                publishProgress("0", null, "Error");
            }

        }
        ips = con.dIps;
        ops = con.dOps;
        publishProgress("0",null,"OK");
```

图 5.12 依据声波通信解析出的 ip 连接 server

5.2 传输模块

在这一模块，进行具体的文件传输。

发送方（即服务器）：

获取到带发送的文件列表，等待客户端请求发送的命令，获取后便开始进行发送文件。

```
@Override
protected Void doInBackground(Void... arg0) {

    try {
        ips = new DataInputStream(socket.getInputStream());
        ops = new DataOutputStream(socket.getOutputStream());
        // 給客戶端發送信息
        MessageUtil.sendMsg("200", ops);

        String msg = MessageUtil.getMsg(ips);
        Log.e("客户端发来的信息", msg);
        System.err.println("已经向客户端发送主机名");

        // 获取命令
        String command = MessageUtil.getMsg(ips);
        if (command.equals(Constant.REQUESTFILES)) {//表示客户端的请求是发送文件
            // 向客户端发送文件
            ArrayList<FileInfo> checkedFiles = AdapterOfGridView_operationComputer
                    .getCheckedFiles();
             ArrayList<FileInfo> fileInfos = AdapterOfGridView_operationComputer.fileInfos;
             fileInfos.addAll(checkedFiles);
            // 先发文件数量
            MessageUtil.sendMsg(fileInfos.size() + "", ops);
            // 发送文件
            for(FileInfo info : fileInfos){
                sendFile(info);
            }
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

图 5.13 发送端发送文件

接收方（即客户端）：

连接服务端成功后，请求服务端（发送者）发送文件，然后获取文件。

```
@Override
protected Void doInBackground(String... value) {
    try {

        InetSocketAddress address = new InetSocketAddress(value[0],
                Constant.PORT);
        Connect con = Connect.getInstance();

        // con.disConnect();
        int times = 0;
        while (!con.connectServer(address, Constant.PORT)) {
            Thread.sleep(100);
            times++;
            if (times > 100) {
                // 协议：第一位是进度百分比，第二位是文件名，第三位是信号
                publishProgress("0", null, "Error");
            }
        }
        ips = con.dIps;
        ops = con.dOps;
        publishProgress("0",null,"OK");
        con.sendMsg(Constant.REQUESTFILES);// 发送一百，表示请求服务器发送文件过来
        // 发来的文件数量
        int count = Integer.parseInt(con.getMsg());
        while (count > 0) {
            receiveFile();
            Log.e("接受文件成功!!!!", "--------------------");
            count--;
        }
        // mAct.mHanlder.sendEmptyMessage(400);
    } catch (Exception e) {
        e.printStackTrace();
        // TODO: handle exception
    }
    return null;

}
```

图 5.14 客户端请求发送文件，读取并处理文件流

5.3 文件管理

提供对文件删改、再次分享等一系列操作

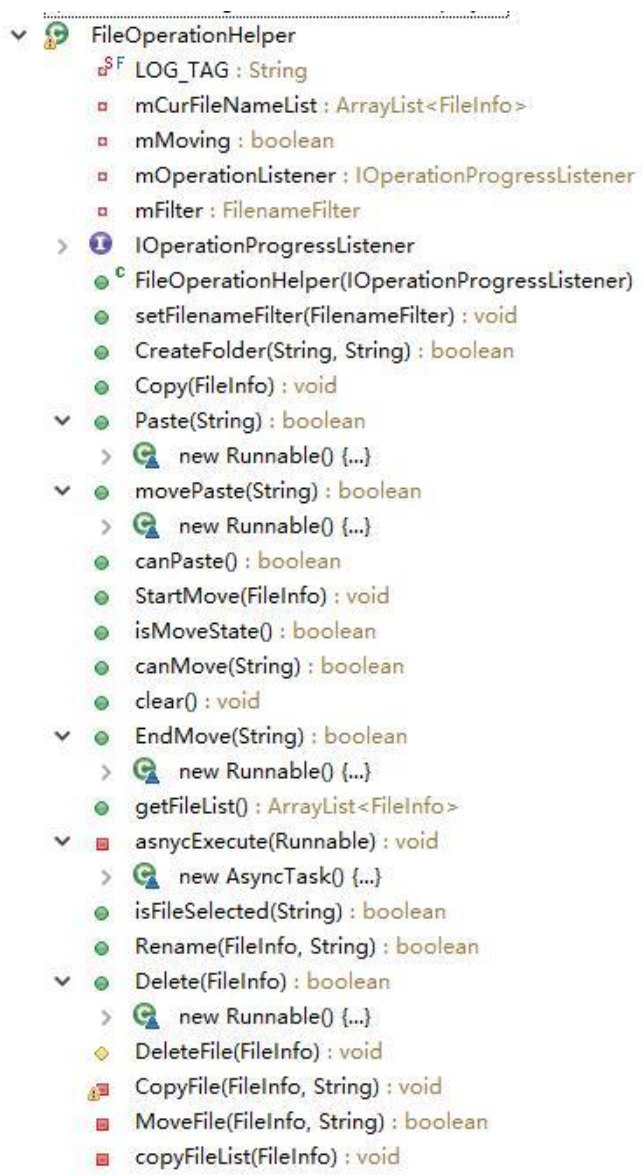


图 5.15 文件管理帮助类

为了针对不同的文件，调用不同的软件进行打开，需要先对文件进行分类。

```
// 获取文件mimetype
static String getMIMEType(File file) {
    String type = "";
    String name = file.getName();
    // 文件扩展名
    String end = name.substring(name.lastIndexOf(".") + 1, name.length())
            .toLowerCase();
    if (end.equals("m4a") || end.equals("mp3") || end.equals("wav")) {
        type = "audio";
    } else if (end.equals("mp4") || end.equals("3gp")) {
        type = "video";
    } else if (end.equals("jpg") || end.equals("png") || end.equals("jpeg")
            || end.equals("bmp") || end.equals("gif")) {
        type = "image";
    } else {
        // 如果无法直接打开，跳出列表由用户选择
        type = "*";
    }
    type += "/*";
    return type;
}
```

图 5.16 文件分类

```
// 打开文件
public static Intent openFile(File file) {
    Intent intent = new Intent();
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    intent.setAction(android.content.Intent.ACTION_VIEW);
    String type = getMIMEType(file);
    intent.setDataAndType(Uri.fromFile(file), type);
    return intent;
}
```

图 5.17 打开文件

5.4 界面

5.3.1 主界面

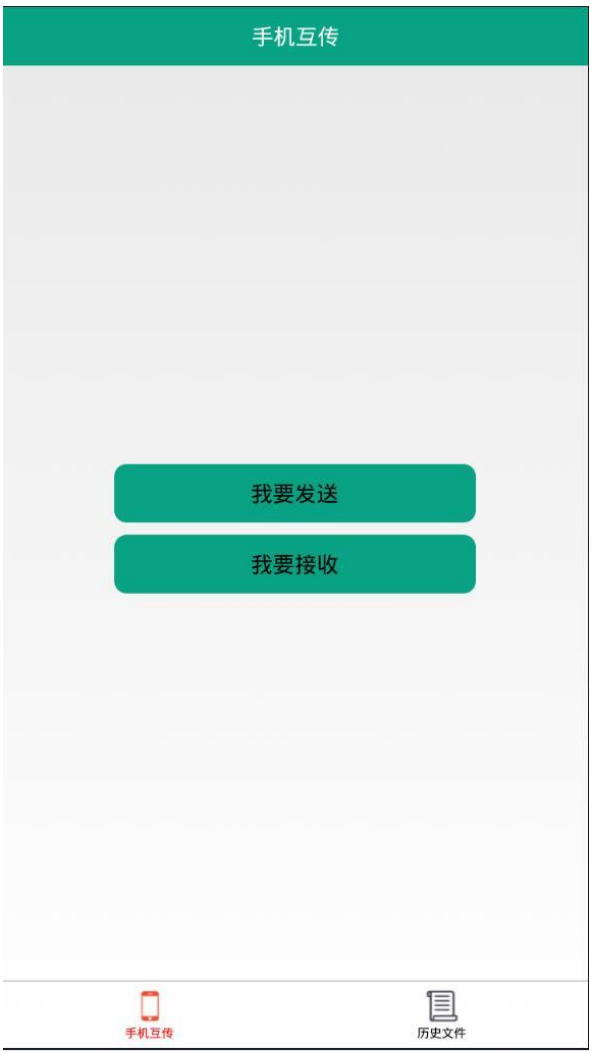


图 5.18 主界面

5.3.2 文件传输

发送方

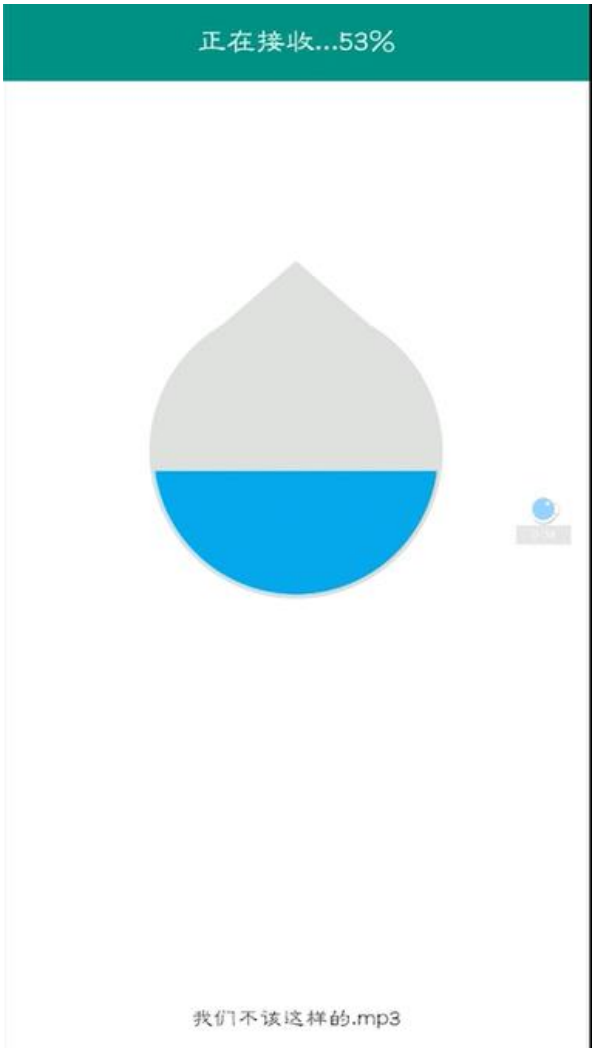
5.19 发送方选择文件

5.20 发送方选择完文件，等待接收方连接

接收方

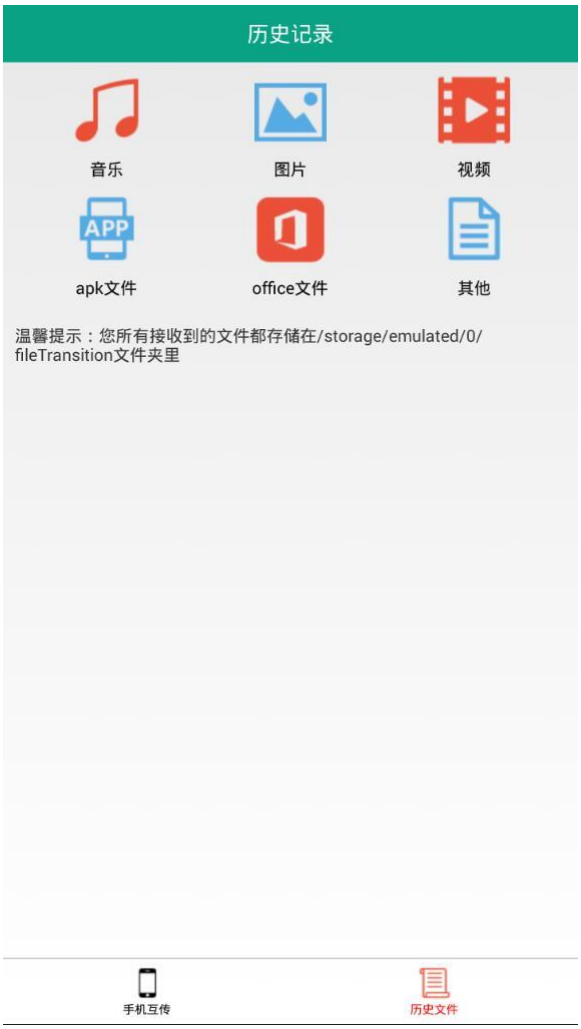
5.21 接收方连接发送方

连接成功后，进入传输页面

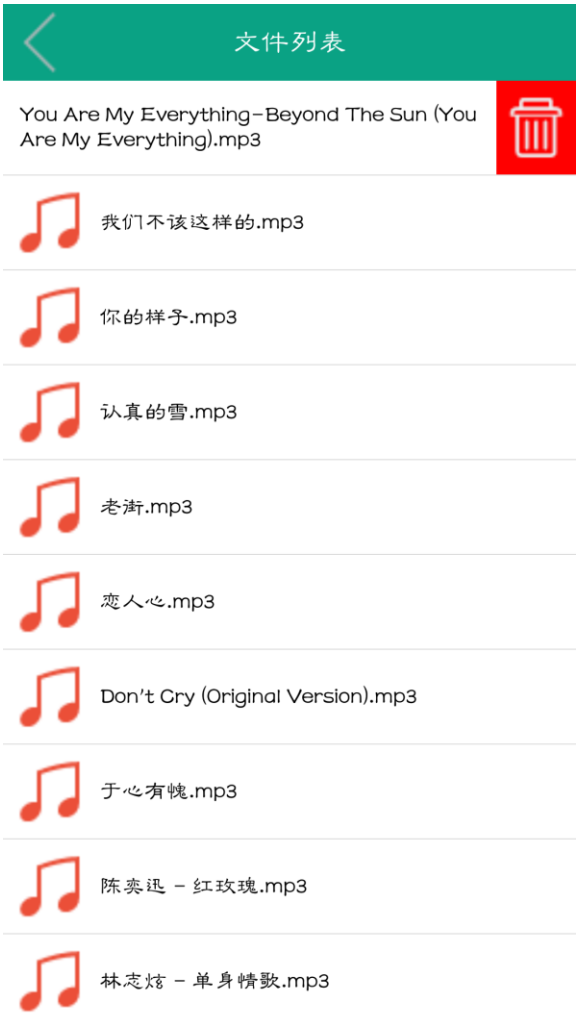


5.22 传输页面

5.3.3 文件管理页面

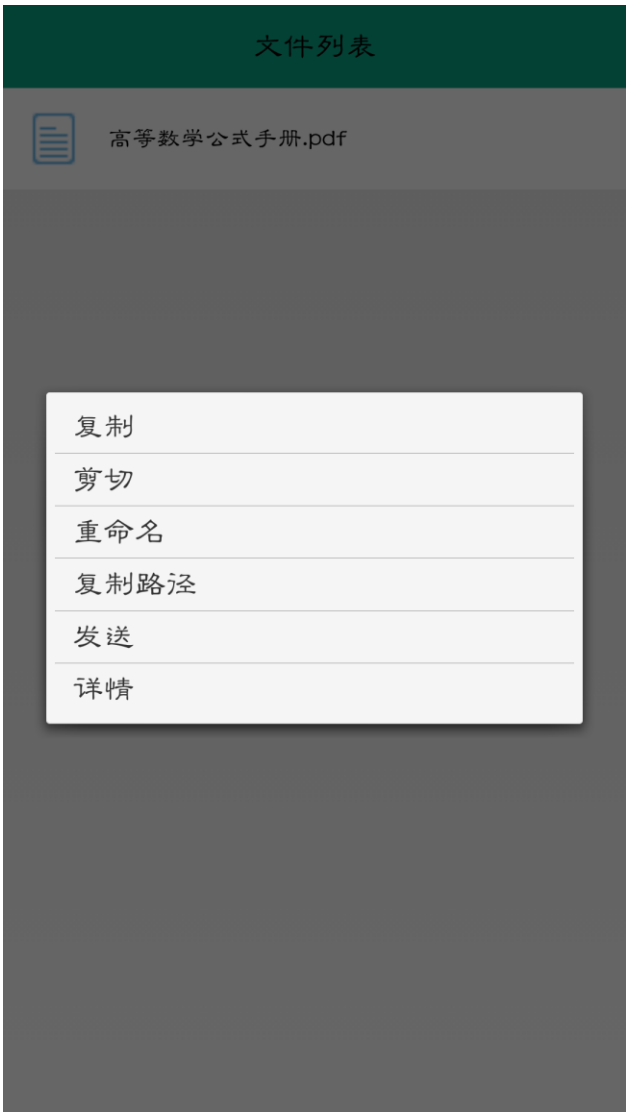


5.23 历史文件管理页面



5.24 侧滑删除

长按，即可唤出文件操作列表



5.25 文件操作列表

5.4 快速文件传递 APP 开发环境

开发工具：eclipse adt

开发平台：Windows 7

开发语言：Java

CPU：i5 4200M

内存：12G

版本控制工具：SVN

手机系统：小米 4，android 5.0

总结和展望

总结

在寻找并储备了接近一年的关于快速文件传输方面的专业知识，经过了这段时间的不懈努力，基于 WiFi 的局域网通讯的 Android 平台下的快速文件传递 APP 终于比较好的实现了。主要创新点有：

1）设计并实现了完整的基于 WiFi 的文件传输功能。通过这种方式进行文件的传输，不仅快速，而且通信的距离相对较远，传输距离理论上可达 100 米。因此，能够适应生活中的多种复杂的传输环境；

2）借助第三方的声波传输的库——SinVoice，传输服务端的 ip，客户端对声波进行录音并解析，解析成功后开始连接服务端。整个操作不需要用户操作，自动地建立 Socket 连接，在人机交互上，优化了交互逻辑，相比传统的蓝牙技术的繁琐的配对操作，增强了用户的体验。

本 APP 为人们在生活、工作、学习等情景中，方便、快速、稳定地传输文件具有极大的意义。首先，文件传输无需通过外网，这解决了目前流量资费较贵的问题，尽管李总理三次提及提速降费，然而实效不大。其次，在传输中，摒弃了传统的蓝牙等传输方式，选择了基于 WiFi 的方式进行文件的传输，这种方式快速、稳定。在当今，摄像头的成像素质越来越优秀，手机的存储空间也越来越大，随之而来的，文件的大小也越来越大，传统的文件传输的方式依然无法满足我们的需求，而基于 WiFi 的方式，在 2.4Ghz 的频段下，最高理论传输速度可达 108Mbps，可以快速、稳定的传输文件。

期间遇到了不少问题，也发现了一些新的技术，也很遗憾由于时间等原因，没能将发现的一些新技术整合到项目中。同时，也为自己遇到问题能做到不放弃，不抛弃，并不断尝试解决方案而感到骄傲。尽管这款 APP 可能仍存在一定的问题、界面等仍有优化的空间，但这不会影响这款 APP 能做到解决用户的痛点这一特质，更不会打消我继续深入学习 Android 开发和网络通信的兴趣。我也很开心，这款 APP 仍存在问题，这能够让我发现我的不足之处，指引并激励我不断的努力学习来弥补

我的不足和完善我的知识体系。

展望

本 APP 主要存在以下几个问题：

1. 在通信上，Google 早在 Android4.0 中便引入了 WiFi Peer-to-Peer API，供开发者使用，能够做到更快速、更高效，在立项时没有了解到，有些遗憾。希望在项目重构时，可以将技术更新为这个。

2. 由于设计不到位，以及时间的关系，我在界面上做的不够优美，没能达到赏心悦目的地步。

在以后的学习、工作中我会继续对新技术，新知识保持一颗渴望的心，不断学习并努力加以运用，希望早日成为理想中的自己。

参考文献

[1] 余志龙，陈小凤.AndroidSDK开发范例大[M].北京：清华大学出版社，2010.

[3] 谢希仁.计算机网络[M].大连：大连理工大学出版社，1996.

[4] 霍斯特曼.JAVA 核心技术卷 1[M].机械工业出版社，2008.

[5] 霍斯特曼.JAVA 核心技术卷 2[M].机械工业出版社，2008.

[6] Dave MacLean.精通Android3[M].北京：人民邮电出版社，2011.

[7] Gourley.HTTP 权威指南[M].北京:人民邮电出版社,2012.9.

[8] 陈文，郭依正.深入理解 Android 网络编程：技术详解与最佳时间[M].北京：机械工业出版社，2013.

[9] 上野·宣.图解HTTP[M].北京.人民邮电出版社,2014-05.

[10] W.Richard Stevens.计算机科学丛书·TCP/IP 详解卷 1：协议[M].机械工业出版社,2014-06.

[11] W.Richard Stevens.计算机科学丛书·TCP/IP 详解卷 2：实现[M].机械工业出版社,2014-06.

[12] W.Richard Stevens.计算机科学丛书·TCP/IP 详解卷 3：TCP 事务协议、HTTP、NNTP 和 UNIX 域协议[M].机械工业出版社,2014-06.

[13] 顾浩鑫.Android高级进阶[M].北京:电子工业出版社，2016.10.

[14] 闫伟，叶建栲.多线程技术在 android 手机开发中的应用[J].信息通信，2012(1):46-47.

[15] 石建华，聂文芳，文晓棠.基于 Android 平台多线程断点续传技术研究[J].科技研究，2012(06):中国电子商务 114-中国电子商务 1151.

[16] 杨杰.基于 Android 的多线程处理技术[J].ISSN 1009-3044,2013:软件设计开发 4251-软件设计开发 4254.

[17] 周兵.基于 Android 的多线程断点下载的研究与实现[N].郧阳师范高等专科学校学报，2012 年 12 月，第 32 卷第 6 期.

[18] Shane Conde.Android Wireless Application Development[M].Addison-Wesley,2010

[19] P.Nevers ,S.Sargento, R.L.Aguiar.Support of Real-Time Services over Integrated 802.16 Metropolitan and Local Area Networks[J].Computers and Communications , 2006.ISCC' 06.Proceedings.11th IEEE Symposium on

[20] Dusit niyato, Ekram Hossain.WIRELESS BROADBAND ACCESS: WIMAX AND BEYOND - Integration of WiMAX and WiFi: Optimal Pricing for Bandwidth Sharing[J].IEEE Communications Magazine,2007(May):Volume:45,Issue:5

[21] 林行高.移动支付应用中的无线通信技术研究[D].华南理工大学，2011

[22] 夏寅昕.大型客机驾驶舱机组行为监测系统设计[D].上海交通大学，2011

[23] 刘夏辰.嵌入式智能家庭网关的设计与实现[D].内蒙古科技大学，2012

[24] 王晓飞.基于 Android 平台移动 OA 的设计与实现[D].北京交通大学，2012

[25] 杨恒畅. Android 平台下浏览器若干功能的设计与实现[D].南京大学，2012

[26] 张庆鑫.基于 Android 平台的手机恶意软件桌面监控系统[D].山东大学，2012

[27] 张瑾.基于 Android 的文件浏览器设计与实现[D].山东大学，2013

[28] 金凌云.基于跨银行现金管理的海关现金管理系统的设计与实现[D].上海交通大学，2013

[29] 左文豪.基于 Android 系统的增强现实技术的研究与实现[D].湖南大学，2013

[30] 沈洋.安全文件服务系统的研究与开发[D].武汉理工大学，2014

[31] 杨艾平.基于 Wi-Fi Direct 的 Android 文件传输研究与实现[D].东华大学，2014

[32] 魏菲.基于 WiFi 定位技术的自助导游系统的研究与设计[D].延边大学，2014

[33] 刘洋.基于人脸识别在教务系统的应用--android 端系统的设计与实现[D].南华大学，2015

[34] https://developer.android.com/index.html

[35] http://www.w3school.com.cn/tcpip/tcpip_intro.asp

[36] www.21ctn.com.news.04-0209/c12.htm

[37] https://zh.wikipedia.org/wiki/%E8%97%8D%E7%89%99

[38] https://zh.wikipedia.org/wiki/%E8%97%8D%E7%89%99

谢 辞

时光飞逝，岁月留痕，转眼间本科四年即将结束了，往昔犹如昨日，历历在目。在这四年中，有太多值得回味、值得追忆、值得珍惜的时光。这份毕业设计凝聚着我本科的所学知识以及将近一年的心血来准备和学习相关知识，并运用到实践中。这份毕设为我的学习生涯交了一份满意的答卷，也是我将来职业生涯的开端。

感谢我的导师余颖老师。从最初论文的选题，需求分析，系统设计，论文的编写等，始终给予我最真切的指导，对我的这一年的学习和生活也做了不少关怀和帮助。在这期间，她花费了大量的宝贵的时间和精力，为我的论文的写作，修改，到最终的定稿做出了很大的指导和实质性的意见。借此机会，向她表示由衷的感谢，此外，她严谨的治学态度，扎实的专业基础，以及耐心的传道授业解惑，使我印象深刻，让我学到了许多为人处世方面的价值观和方法，必将是我受益终身。

感谢四年来给我关心和支持的各位老师，给我提供帮助的各位学长学姐，他们不仅交给我许许多多的专业知识，还教会了我面对问题不恐惧，用于正视问题，学习分析问题、解决问题的方法以及独立思考，开拓创新的能力。在实现本系统时，之前与我一起参加比赛的同学给予了我很大的帮助，尤其是吴洪全同学，经常深更半夜一起讨论，非常感谢。借此机会，对大家表示衷心的感谢！

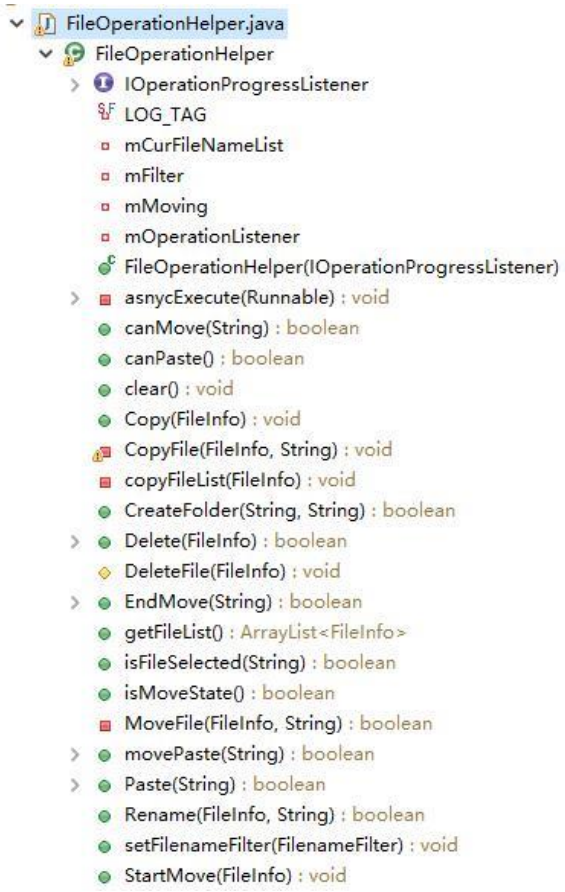
感谢在校期间，有一个这样的团队——Geowind，在这里学生自主管理，我学会了独立；在这里学长带领学弟，给了我很大的学习帮助和成长的空间;在这里，经常关注一些赛事，与小伙伴一起组队参赛，让我成长了许多，见识了许多，还结交了许多志同道合的朋友，非常感谢！感谢在帆软的实习经历，给了我许多锻炼和学习的机会，感谢我的同事，帮助我克服一个又一个困难！

最后要感谢背后一直默默支持我的父母，是您们让我无忧无虑的学习和生活，是我的精神支柱，无论发生了什么，都会帮我排忧解难，在此，深深感谢！

最后，向今天参与我论文评审的各位老师致以深深的敬意，谢谢你们的辛勤工作！

附 录

附录 1 文件操作帮助类的代码



复制、粘贴、删除等一系列文件操作

```
public class FileOperationHelper {
    private static final String LOG_TAG = "FileOperation";
    private ArrayList<FileInfo> mCurFileNameList = new ArrayList<FileInfo>();
    private boolean mMoving;
    private IOperationProgressListener mOperationListener;
    private FilenameFilter mFilter;

    public interface IOperationProgressListener {
```

```
    void onFinish();
    void onFileChanged(String path);
}

public FileOperationHelper(IOperationProgressListener l) {
    mOperationListener = l;
}

public void setFilenameFilter(FilenameFilter f) {
    mFilter = f;
}

public boolean CreateFolder(String path, String name) {
    Log.v(LOG_TAG, "CreateFolder >>> " + path + "," + name);

    File f = new File(Util.makePath(path, name));
    if (f.exists())
        return false;
    return f.mkdir();
}

public void Copy(FileInfo files) {
    copyFileList(files);
}

/**
 * 复制
 * @param path
```

```
     * @return 成功与否
     */
    public boolean Paste(String path) {
        if (mCurFileNameList.size() == 0) {
            if (App.map_file_application.size() == 0) {
                return false;
            } else {

mCurFileNameList.add(App.map_file_application.get("file"));
            }
        }

        final String _path = path;
        asnycExecute(new Runnable() {
            @Override
            public void run() {
                for (FileInfo f : mCurFileNameList) {
                    CopyFile(f, _path);
                }
                mOperationListener.onFileChanged(Environment
                .getExternalStorageDirectory().getAbsolutePath());
                clear();
            }
        });
        return true;
    }

    public boolean movePaste(String path) {
```

```
            if (mCurFileNameList.size() == 0) {
                if (App.map_file_application.size() == 0) {
                    return false;
                } else {
    mCurFileNameList.add(App.map_file_application.get("file"));
                }
            }

            final String _path = path;
            asnycExecute(new Runnable() {
                @Override
                public void run() {
                    for (FileInfo f : mCurFileNameList) {
                        CopyFile(f, _path);
                    }
        mOperationListener.onFileChanged(Environment.getExternalStorageDi
rectory().getAbsolutePath());
                    clear();
                }
            });
            return true;
        }

        public boolean canPaste() {
            if (mCurFileNameList.size() == 0) {
                return App.map_file_application.size() != 0;
            } else {
                return mCurFileNameList.size() != 0;
```

```
    }
}

public void StartMove(FileInfo files) {
    if (mMoving)
        return;
    mMoving = true;
    copyFileList(files);
}

public boolean isMoveState() {
    return mMoving;
}

public boolean canMove(String path) {
    for (FileInfo f : mCurFileNameList) {
        if (!f.IsDir)
            continue;
        if (Util.containsPath(f.filePath, path))
            return false;
    }
    return true;
}

public void clear() {
    synchronized (mCurFileNameList) {
        App.map_file_application.clear();
        mCurFileNameList.clear();
```

```
    }
}

public boolean EndMove(String path) {
    if (!mMoving)
        return false;
    mMoving = false;
    if (TextUtils.isEmpty(path))
        return false;
    final String _path = path;
    asnycExecute(new Runnable() {
        @Override
        public void run() {
            for (FileInfo f : mCurFileNameList) {
                MoveFile(f, _path);
            }
            mOperationListener.onFileChanged(Environment
    .getExternalStorageDirectory().getAbsolutePath());
            clear();
        }
    });
    return true;
}

public ArrayList<FileInfo> getFileList() {
    return mCurFileNameList;
}
```

```
private void asnycExecute(Runnable r) {
    final Runnable _r = r;
    new AsyncTask<Void,Void,Void>() {
        @Override
        protected Void doInBackground(Void... params) {
            synchronized (mCurFileNameList) {
                _r.run();
            }
            if (mOperationListener != null) {
                mOperationListener.onFinish();
            }
            return null;
        }

    }.execute();
}

public boolean isFileSelected(String path) {
    synchronized (mCurFileNameList) {
        for (FileInfo f : mCurFileNameList) {
            if (f.filePath.equalsIgnoreCase(path))
                return true;
        }
    }
    return false;
}

public boolean Rename(FileInfo f, String newName) {
```

```
            if (f == null || newName == null) {
                Log.e(LOG_TAG, "Rename: null parameter");
                return false;
            }
            File file = new File(f.filePath);
            String newPath =
Util.makePath(Util.getPathFromFilepath(f.filePath),
                    newName);
            final boolean needScan = file.isFile();
            try {
                boolean ret = file.renameTo(new File(newPath));
                if (ret) {
                    if (needScan) {
                        mOperationListener.onFileChanged(f.filePath);
                    }
                    mOperationListener.onFileChanged(newPath);
                }
                return ret;
            } catch (SecurityException e) {
                Log.e(LOG_TAG, "Fail to rename file," + e.toString());
            }
            return false;
        }

        public boolean Delete(FileInfo files) {
            copyFileList(files);
            asnycExecute(new Runnable() {
                @Override
```

```
            public void run() {
                for (FileInfo f : mCurFileNameList) {
                    DeleteFile(f);
                }
                mOperationListener.onFileChanged(Environment
.getExternalStorageDirectory().getAbsolutePath());
                clear();
            }
        });
        return true;
    }

    protected void DeleteFile(FileInfo f) {
        if (f == null) {
            Log.e(LOG_TAG, "DeleteFile: null parameter");
            return;
        }

        File file = new File(f.filePath);
        boolean directory = file.isDirectory();
        if (directory) {
            for (File child : file.listFiles(mFilter)) {
                if (Util.isNormalFile(child.getAbsolutePath())) {
                    DeleteFile(Util.GetFileInfo(child, mFilter, true));
                }
            }
        }
        file.delete();
```

```
            Log.v(LOG_TAG, "DeleteFile >>> " + f.filePath);
        }

        private void CopyFile(FileInfo f, String dest) {
            if (f == null || dest == null) {
                Log.e(LOG_TAG, "CopyFile: null parameter");
                return;
            }

            File file = new File(f.filePath);
            if (file.isDirectory()) {
                // directory exists in destination, rename it
                String destPath = Util.makePath(dest, f.fileName);
                File destFile = new File(destPath);
                int i = 1;
                while (destFile.exists()) {
                    destPath = Util.makePath(dest, f.fileName + " " + i++);
                    destFile = new File(destPath);
                }

                for (File child : file.listFiles(mFilter)) {
                    if (!child.isHidden()
                            && Util.isNormalFile(child.getAbsolutePath())) {
                        CopyFile(Util.GetFileInfo(child, mFilter, Settings
                                .instance().getShowDotAndHiddenFiles()),
destPath);
                    }
                }
```

```
        } else {
            String destFile = Util.copyFile(f.filePath, dest);
        }
        Log.v(LOG_TAG, "CopyFile >>> " + f.filePath + "," + dest);
    }

    private boolean MoveFile(FileInfo f, String dest) {
        Log.v(LOG_TAG, "MoveFile >>> " + f.filePath + "," + dest);
        if (f == null || dest == null) {
            Log.e(LOG_TAG, "CopyFile: null parameter");
            return false;
        }
        File file = new File(f.filePath);
        String newPath = Util.makePath(dest, f.fileName);
        try {
            return file.renameTo(new File(newPath));
        } catch (SecurityException e) {
            Log.e(LOG_TAG, "Fail to move file," + e.toString());
        }
        return false;
    }

    private void copyFileList(FileInfo files) {
        synchronized (mCurFileNameList) {
            mCurFileNameList.clear();
            App.map_file_application = new HashMap<String, FileInfo>();
            App.map_file_application.put("file", files);
            App.map_file_application.size();
```

```
                mCurFileNameList.add(files);
            }
        }
    }
```

附录 2 热点管理的代码

```
public class AccessPointManager extends WifiManagerWrap {
    private static int mWifiApStateDisabled;
    private static int mWifiApStateDisabling;
    private static int mWifiApStateEnabled;
    private static int mWifiApStateEnabling;
    private static int mWifiApStateFailed;
    private static String mWifiApStateChangedAction;
    private static String mExtraWifiApState;
    private static String mExtraPreviousWifiApState;
    private String mWifiApSSID;
    static {
        try {
            mWifiApStateDisabled = WifiManager.class.getField(
                "WIFI_AP_STATE_DISABLED").getInt(WifiManager.class);
            mWifiApStateDisabling = WifiManager.class.getField(
                "WIFI_AP_STATE_DISABLING").getInt(WifiManager.class);
            mWifiApStateEnabled = WifiManager.class.getField(
                "WIFI_AP_STATE_ENABLED").getInt(WifiManager.class);
            mWifiApStateEnabling = WifiManager.class.getField(
                "WIFI_AP_STATE_ENABLING").getInt(WifiManager.class);
            mWifiApStateFailed = WifiManager.class.getField(
                "WIFI_AP_STATE_FAILED").getInt(WifiManager.class);
```

```
                mWifiApStateChangedAction = (String)
                WifiManager.class.getField(
                    "WIFI_AP_STATE_CHANGED_ACTION").get(WifiManager.class);
                mExtraWifiApState = (String) WifiManager.class.getField(
                    "EXTRA_WIFI_AP_STATE").get(WifiManager.class);
                mExtraPreviousWifiApState = (String)
                WifiManager.class.getField(
                    "EXTRA_PREVIOUS_WIFI_AP_STATE").get(WifiManager.class);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        /** wifi hot spot is disabled */
        public static final int WIFI_AP_STATE_DISABLED =
mWifiApStateDisabled;
        /** wifi hot spot is disabling */
        public static final int WIFI_AP_STATE_DISABLING =
mWifiApStateDisabling;
        /** wifi hot spot is enabled */
        public static final int WIFI_AP_STATE_ENABLED = mWifiApStateEnabled;
        /** wifi hot spot is enabling */
        public static final int WIFI_AP_STATE_ENABLING =
mWifiApStateEnabling;
        /** wifi hot spot state : failed */
        public static final int WIFI_AP_STATE_FAILED = mWifiApStateFailed;
        /** wifi hot spot state changed */
        public static final String WIFI_AP_STATE_CHANGED_ACTION =
```

```
mWifiApStateChangedAction;
        /** wifi hot spot state */
        public static final String EXTRA_WIFI_AP_STATE = mExtraWifiApState;
        /** the former wifi hot spot state */
        public static final String EXTRA_PREVIOUS_WIFI_AP_STATE =
mExtraPreviousWifiApState;

        private OnWifiApStateChangeListener mOnApStateChangeListener;
        private BroadcastReceiver mWifiApStateReceiver = new
BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                String action = intent.getAction();

                if (WIFI_AP_STATE_CHANGED_ACTION.equals(action)
                        && mOnApStateChangeListener != null) {

    mOnApStateChangeListener.onWifiStateChanged(intent.getIntExtra(
                            EXTRA_WIFI_AP_STATE, WIFI_AP_STATE_FAILED));
                }
            }
        };

        public AccessPointManager(Context context) {
            super(context);
            if (!TextUtils.isEmpty(WIFI_AP_STATE_CHANGED_ACTION)) {
                IntentFilter intentFilter = new IntentFilter();
                intentFilter.addAction(WIFI_AP_STATE_CHANGED_ACTION);
```

```
            context.registerReceiver(mWifiApStateReceiver,
intentFilter);
        } else {
        }
    }

    /** set listener to monitor the state of access point */
    public void setWifiApStateChangeListener(
            OnWifiApStateChangeListener listener) {
        mOnApStateChangeListener = listener;
    }

    /** set WiFi hot spot is enableed */
    private boolean setWifiApEnabled(WifiConfiguration configuration,
            boolean enabled) {
        try {
            if (enabled) {
                mWifiManager.setWifiEnabled(false);
            }

            Method method = mWifiManager.getClass().getMethod(
                    "setWifiApEnabled",          WifiConfiguration.class,
Boolean.TYPE);
            return (Boolean) method
                    .invoke(mWifiManager, configuration, enabled);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
            return false;
        }

        /** get WiFi hot spot state */
        public int getWifiApState() {
            try {
                Method                          method                          =
mWifiManager.getClass().getMethod("getWifiApState");
                return (Integer) method.invoke(mWifiManager);
            } catch (Exception e) {
                e.printStackTrace();
            }

            return mWifiApStateFailed;
        }

        /**
         * 鑾峰彇Wifi鎺夯倍閰嶇疆
         *
         * @return WifiConfiguration
         */
        private WifiConfiguration getWifiApConfiguration() {
            try {
                Method method = mWifiManager.getClass().getMethod(
                        "getWifiApConfiguration");
                WifiConfiguration config = (WifiConfiguration) method
                        .invoke(mWifiManager);
```

```
            loadWifiConfigurationFromProfile(config);
            return config;
        } catch (Exception e) {
            e.printStackTrace();
        }

        return null;
    }

    @SuppressWarnings("unused")
    private boolean setWifiApConfiguration(WifiConfiguration
configuration) {
        try {
            Method method = mWifiManager.getClass().getMethod(
                    "setWifiApConfiguration", WifiConfiguration.class);
            return (Boolean) method.invoke(mWifiManager, configuration);
        } catch (Exception e) {
            e.printStackTrace();
        }

        return false;
    }

    /**
     * wifi hot spot is enabled?
     *
     * @return wifi 鏂戛倍鏄●悊鏈夋晲
     */
```

```
public boolean isWifiApEnabled() {
    try {
        Method method = mWifiManager.getClass()
                .getMethod("isWifiApEnabled");
        return (Boolean) method.invoke(mWifiManager);
    } catch (Exception e) {
        e.printStackTrace();
    }

    return false;
}

/** create WiFi Hot Spot SSID suffix */
public void createWifiApSSID(String suffix) {
    mWifiApSSID = DEFAULT_SSID + suffix;
}

/** create WiFi hot spot */
public boolean startWifiAp() {
    // close WiFi
    // if(mWifiManager.isWifiEnabled()){
    closeWifi();
    // close the open WiFi hot spot
    if (DeviceUtils.isHTC()) { // 2013-12-31 add special for htc
        enableHtcHotspot(getWifiApConfiguration(), false);
        return setHtcHotspot();
    }
```

```
    if (isWifiApEnabled())
        setWifiApEnabled(getWifiApConfiguration(), false);
    // envoke the hot spot need to be created
    acquireWifiLock();
    WifiConfiguration wifiConfig = new WifiConfiguration();
    // 手机型号
    // String model = android.os.Build.MODEL + "_ap";
    createWifiApSSID("whqtest"); // ap 名称
    wifiConfig.SSID = getWifiApSSID();

    // wifiConfig.preSharedKey = DEFAULT_PASSWORD;
    // setWifiConfigAsWPA(wifiConfig); // have password
    setWifiConfig(wifiConfig); // no password
    return setWifiApEnabled(wifiConfig, true);
}

/**
 * connect the ap
 *
 * @param SSID
 *            the ap's ssid
 * @param password
 *            the ap's sharedKey,null if not need password
 * @return true if connect success,false otherwise;
 */
public boolean connectAP(String SSID, String password) {

    openWifi();
```

```
        // if it had connected ,stop connect
        stopWifi();

        WifiConfiguration config = new WifiConfiguration();
        config.SSID = "\"" + SSID  + "\"" ;

        if (password != null) {
            config.preSharedKey = "\"" +  password + "\"" ;//主意格式
        }
        config.hiddenSSID = true;
        config.status = WifiConfiguration.Status.ENABLED;

config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.TKIP);

config.allowedGroupCiphers.set(WifiConfiguration.GroupCipher.CCMP);
 //
config.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.WPA_PSK);
        config.allowedPairwiseCiphers
                .set(WifiConfiguration.PairwiseCipher.TKIP);
        config.allowedPairwiseCiphers
                .set(WifiConfiguration.PairwiseCipher.CCMP);
        config.allowedProtocols.set(WifiConfiguration.Protocol.RSN);

        int netId = mWifiManager.addNetwork(config);

        return mWifiManager.enableNetwork(netId, true);

    }
```

```
        /** set HTC WiFi hotspot parameter, 2013-12-31 add special for htc
*/
        private boolean setHtcHotspot() {

            WifiConfiguration apConfig = new WifiConfiguration();

            apConfig.SSID = getWifiApSSID();

            // set encrypt type
            apConfig.wepKeys[0] = null;

    apConfig.allowedKeyManagement.set(WifiConfiguration.KeyMgmt.NONE);
            apConfig.wepTxKeyIndex = 0;

            if (!setHTCSSID(apConfig))
                return false;

            boolean flag = enableHtcHotspot(apConfig, true);

            return flag;
        }

        /** enable and disable htc WiFi hotspot, 2013-12-31 add special for
htc */
        private boolean enableHtcHotspot(WifiConfiguration apConfig,
boolean enable) {
```

```
            Method method = null;
            try {
                method                                                    =
mWifiManager.getClass().getMethod("setWifiApEnabled",
                        WifiConfiguration.class, Boolean.TYPE);
            } catch (NoSuchMethodException e) {
                e.printStackTrace();
            }

            boolean flag = false;
            try {
                flag  =  (Boolean)  method.invoke(mWifiManager,  apConfig,
enable);
            } catch (IllegalArgumentException e) {
                flag = false;
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                flag = false;
                e.printStackTrace();
            } catch (InvocationTargetException e) {
                flag = false;
                e.printStackTrace();
            }

            return flag;
        }

        /** set htc hot spot SSID etc. 2013-12-31 add special for htc */
```

```
        public boolean setHTCSSID(WifiConfiguration config) {
            Field localField1;
            boolean successed = true;
            try {
                localField1 = WifiConfiguration.class
                        .getDeclaredField("mWifiApProfile");
                localField1.setAccessible(true);
                Object localoObject1 = localField1.get(config);
                localField1.setAccessible(false);

                if (localoObject1 != null) {
                    Field                    localField2                    =
localoObject1.getClass().getDeclaredField(
                            "SSID");
                    localField2.setAccessible(true);
                    localField2.set(localoObject1, config.SSID);
                    localField2.setAccessible(false);

                    Field                    localField4                    =
localoObject1.getClass().getDeclaredField(
                            "BSSID");
                    localField4.setAccessible(true);
                    localField4.set(localoObject1, config.BSSID);
                    localField4.setAccessible(false);

                    Field localField3 =
    localoObject1.getClass().getDeclaredField("dhcpEnable");
                    localField3.setAccessible(true);
```

```
                localField3.set(localoObject1, 1);
                localField3.setAccessible(false);



                Field locaField4 =
localoObject1.getClass().getDeclaredField("secureType");
                locaField4.setAccessible(true);
                locaField4.set(localoObject1, "open"); // very important,
no
                                                    // encrypt, must open
                locaField4.setAccessible(false);

            }
        } catch (Exception e) {
            successed = false;
        }
        return successed;
    }

    /** stop WiFi Hot spot and refresh the former WiFi state */
    public void stopWifiAp() {
        releaseWifiLock();
        // close WiFi hot spot
        if (isWifiApEnabled())
            setWifiApEnabled(getWifiApConfiguration(), false);
        else
            enableHtcHotspot(getWifiApConfiguration(), false);
```

```
}

/** stop WiFi AP and start original state */
public void stopWifiAp(boolean isWiFiActive) {
    releaseWifiLock();
    // close WiFi hot spot
    if (isWifiApEnabled())
        setWifiApEnabled(getWifiApConfiguration(), false);
    else
        enableHtcHotspot(getWifiApConfiguration(), false);

    if (isWiFiActive)
        openWifi();
}


public void destroy(Context context) {
    stopWifiAp();
    removeNetwork(SSID_PREFIX);
    context.unregisterReceiver(mWifiApStateReceiver);
}

/**
 * get access point ssid
 *
 * @return ssid string
 */
public String getWifiApSSID() {
```

```
        return mWifiApSSID;
    }

    /** get WiFi hot spot gate */
    public String getWiFiHotSpotGate() {
        String gateString = null;
        try {
            gateString = getLocalIpAddress();
            if (gateString == null)
                gateString = Constant.FREE_SERVER;
        } catch (UnknownHostException e) {
            e.printStackTrace();
        }

        return gateString;
    }

    /** scan all net Adapter to get all IP, just return 192 */
    public static String getLocalIpAddress() throws UnknownHostException
{

        try {
            for (Enumeration<NetworkInterface> en = NetworkInterface
                    .getNetworkInterfaces(); en.hasMoreElements();) {
                NetworkInterface intf = en.nextElement();
                for (Enumeration<InetAddress> enumIpAddr = intf
                        .getInetAddresses();
enumIpAddr.hasMoreElements();) {
```

```
                    InetAddress inetAddress = enumIpAddr.nextElement();
                    if (!inetAddress.isLoopbackAddress()
                            && (inetAddress.getAddress().length == 4)
                            && inetAddress.getHostAddress().startsWith(
                                    "192.168")) {
                        return inetAddress.getHostAddress();
                    }

                }
            }
        } catch (SocketException ex) {
        }
        return null;
    }

    /**
     * wifi AP
     */
    public static interface OnWifiApStateChangeListener {
        /**
         * access point status listener
         *
         * @param state
         *            wifi state now
         */
        public void onWifiStateChanged(int state);
    }
}
```

附录 3 发送声波的代码

```
private final static int[] TOKENS = { 32, 32, 32, 32, 32, 32 };
private final static int TOKEN_LEN = TOKENS.length;


    private void sendAddress() {
        try {
            String localIpAddress = getLocalIpAddress();
            Log.e("本服务器的 ip", localIpAddress);
            StringBuilder ssid = new StringBuilder();

            ssid.append("whqtest").append(":"); // 文件头信息
            // 加上文件头信息
            ssid.append(localIpAddress);
            Toast.makeText(App.context, localIpAddress, Toast.LENGTH_SHORT)
                    .show();
            byte[] strs = ssid.toString().getBytes();

            if (null != strs) {
                int len = strs.length;
                int[] tokens = new int[len];
                int maxEncoderIndex = mSinVoicePlayer.getMaxEncoderIndex();
                LogHelper.d(TAG, "maxEncoderIndex:" + maxEncoderIndex);
                String encoderText = ssid.toString();
                for (int i = 0; i < len; ++i) {
                    if (maxEncoderIndex < 255) {
                        tokens[i] = Common.DEFAULT_CODE_BOOK
                                .indexOf(encoderText.charAt(i));
                    } else {
```

```
                    tokens[i] = strs[i];
                }
            }
            mSinVoicePlayer.play(tokens, len, false, 2000);
        } else {
            mSinVoicePlayer.play(TOKENS, TOKEN_LEN, false, 2000);
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}

//在 onResume()方法里发送
@Override
    protected void onResume() {
        super.onResume();
        mSinVoicePlayer = new SinVoicePlayer();
        mSinVoicePlayer.init(this);
        mSinVoicePlayer.setListener(this);
        int time = 0;
        while (!accessPoint.isWifiApEnabled()) {// 如果 WiFi 没有被允许使用，就进入循环，等待用户允许；在规定时间内，被允许后，则跳出循环
        // 堵塞
        try {
            time++;
            if (time > 10)
                return;
```

```
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        // 跳出循环，即用户允许了
        sendAddress(); // 启动完成后，把本机 ip 和服务器端口号用声波发送
```

附录 4 接收声波的代码

```
    @Override
    public void onResume() {
        super.onResume();

        mWakeLock.acquire();
        // 接收
        receiveVoice();
    }


/**
     * 接收数据的方法
     *
     * @return
     */
    private String receiveVoice() {
        mRecognition.start(TOKEN_LEN, mIsReadFromFile);
        return null;
    }
```

```
    public void start(final int tokenCount, final boolean isReadFromFile)
{
        if (STATE_STOP == mState) {
            mState = STATE_START;

            mBufferQueue.set();

            mRecognitionThread = new Thread() {
                @Override
                public void run() {
                    mRecognition.start(tokenCount);
                }
            };
            if (null != mRecognitionThread) {
                mRecognitionThread.start();
            }

            mRecordThread = new Thread() {
                @Override
                public void run() {
                    mRecord.start(isReadFromFile);
                }
            };
            if (null != mRecordThread) {
                mRecordThread.start();
            }
        }
    }
```

附录 5 解析 ip 的代码

```
    if (activity.mRecgCount > 0) {
                    byte[] strs = new byte[activity.mRecgCount];
                    for (int i = 0; i < activity.mRecgCount; ++i) {
                        strs[i] = (byte) activity.mRecgs[i];
                    }
                    try {
                        String strReg = new String(strs, "UTF8");
                        Toast.makeText(App.context,                strReg,
Toast.LENGTH_LONG)
                                .show();
                        if (null != mAct) {
                            // 得到 ip 后，链接对方的服务器
                            Log.e("接收到的 IP： ", strReg);
                            // 校验
                            String[] code = strReg.split(":"); // 文件头
信息
                            String ssid = null;
                            String ip = null;

                            if (code.length > 1) {
                                ssid = code[0]; // ssid
                                ip = code[1]; // 正文
                            }
                            // 链接

   activity.wifiConnect.connectToAccessPoint(ssid,
                                    null);
```

```
                        while
(!activity.mWifiManager.getConnectionInfo()
                                .getSSID().contains(ssid)) {
                            Thread.sleep(50);
                            // 等待链接好 ap
                        }
                        activity.mHanlder.sendEmptyMessage(200);
                        // 启动 asyncTask
                        // ap 的地址是写死的,就是下面这个地址
                        new
ClientThread(frame).execute("192.168.43.1");
                    }

                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
```

附录 6 发送文件的代码

```
@Override
protected Void doInBackground(Void... arg0) {


    try {
        ips = new DataInputStream(socket.getInputStream());
        ops = new DataOutputStream(socket.getOutputStream());
        // 获取命令
        String command = MessageUtil.getMsg(ips);
        if (command.equals(Constant.REQUESTFILES)) {//表示客户端的
请求是发送文件
            // 向客户端发送文件
            ArrayList<FileInfo>          checkedFiles          =
AdapterOfGridView_operationComputer
                    .getCheckedFiles();
             ArrayList<FileInfo>           fileInfos           =
AdapterOfGridView_operationComputer.fileInfos;
             fileInfos.addAll(checkedFiles);
            // 先发文件数量
            MessageUtil.sendMsg(fileInfos.size() + "", ops);
            // 发送文件
            for(FileInfo info : fileInfos){
                sendFile(info);
            }
        }


    } catch (Exception e) {
        e.printStackTrace();
```

```
        }
        return null;
    }
```

附录 7 读取文件流的代码

```
    @Override
        protected Void doInBackground(String... value) {
            try {

                InetSocketAddress address = new InetSocketAddress(value[0],
                        Constant.PORT);
                Connect con = Connect.getInstance();

                // con.disConnect();
                int times = 0;
                while (!con.connectServer(address, Constant.PORT)) {
                    Thread.sleep(100);
                    times++;
                    if (times > 100) {
                        // 协议： 第一位 是进度百分比，第二位是文件名，第三
位是信号
                        publishProgress("0", null, "Error");
                    }
                }
                ips = con.dIps;
                ops = con.dOps;
                publishProgress("0",null,"OK");
                con.sendMsg(Constant.REQUESTFILES);// 发送一百， 表示请求服
```

务器发送文件过来

```
            // 发来的文件数量
            int count = Integer.parseInt(con.getMsg());
            while (count > 0) {
                receiveFile();
                Log.e("接受文件成功!!!! ", "--------------------");
                count--;
            }
            // mAct.mHanlder.sendEmptyMessage(400);
        } catch (Exception e) {
            e.printStackTrace();
            // TODO: handle exception
        }
        return null;

    }
```