1. 附近的人：

通过摇一摇上传自己的位置信息（经纬度）到服务器，服务器查找半小时内在一公里范围内分享上传自己的位置信息的陌生人，并把附近的人信息回传到请求客户端，客户端再把附近的人显示出来。整个过程信息交互采用xml数据包进行传递，具体信息包如下所示：

请求包：

<iq id=”” from=”” to=”” type=”get” >

<query xmlns = "urn:xmpp:rayo:lbsservice" >

<item username=”” updatetime=”” gender=”” nickname=”” longitude = "" latitude = "" />

</query>

</iq>

其中<iq>结点是XMPP协议中的IQ数据包头结点，设置它的子节点为<query> ，命名空间为:urn:xmpp:rayo:lbsservice，提供让服务器端识别该数据包，并流向附近的人插件，<query>的子节点<item>为位置信息主体，上传该位置信息的用户，性别，昵称，更新时间以及经纬度，提供这些数据给服务器查找附近的陌生人。

服务器端接收到这个数据包后，查找位置信息数据库，查找到半小时内上传过位置信息的一公里范围内的陌生人，并把附近的人信息回传到请求客户端：

回传IQ数据包信息如下：

<iq id=”” from=”” to=”” type=”result” >

<query xmlns = "urn:xmpp:rayo:lbsservice" >

<item username=”” updatetime=”” gender=”” nickname=”” longitude = "" latitude = ""

jid=””/>

</query>

</iq>

username：用户名

updatetime：上传位置信息的时间

gender：用户性别

nickname：用户昵称

longitude ：经度

latitude ：纬度

jid：用户XMPP唯一的JID

信息内容和发送数据包类似，多了一个XMPP内部的用户专有JID，用来实现添加该好友的功能，其中每个<item>结点表示一个附近的人，回传信息包可以有多个<item>包。

客户端接收到回传的位置信息之后，开启百度地图界面，把收到的附近的人根据经纬度显示在地图上。用户点击地图上的一个陌生人，便会根据收到的JID向服务器请求该陌生人的详细信息，在UI线程上，把界面跳转到解谜游戏界面，等用户把解密游戏攻关之后，便可以把后台请求加载好的详细信息显示在陌生人用户详情上，用户可以选择是否添加为好友。

1. 发送语音信息，分享图片

用户聊天界面下发送语音信息即是软件开启Android的录音模块，把语音信息录音，再通过服务器把语音文件传送到对方，对方点击播放语音即实现了语音发送的功能。

用户聊天界面下，发送图片即是把图片文件通过服务器发送到对方。

综上，唯一的技术难题便是文件传输。

Asmack的文件传输功能有一定的缺陷，导致Android客户端难以实现文件传输功能，于是我们便设想出三种实现文件传输的方式：

1、将传输的文件转换成Base64编码字符串，并把字符串通过XMPP的Message数据包像普通的聊天消息传送给对方，在这种Message上进行标记即可实现此功能。但是这种方式有个致命的问题即是允许传输的单个文件大小限制较小，在传输一些大图片无法实现。

1. 模仿XMPP原本的文件传输的协议，自行开发出一个Openfire的文件传输代理，通过Openfire实现双方的文件信息交互，再通过一端开启一个ServerSocket充当临时文件服务器，由发送端开启一个Socket把文件传送过去。这个方式的弊病即是开启的Socket难以被Openfire服务器调控，文件传输过程中出现网络问题导致的丢包等问题难以被服务器调控保证接收方收到的文件的可靠性。
2. 模仿XMPP的原本的文件传输协议，由服务器代理通过Message实现文件的基本信息的双方协商，再由发送方通过HTTP协议将文件传输到HTTP服务器，并把文件信息告知接收方，接收方通过这个文件信息连接HTTP服务器即可找到对方发送的文件并进行下载。下载完成后HTTP服务器会将该文件删除，保证服务器的空间。这个方式使得服务器在文件传输过程中可以及时检测传输文件的可靠性，并可在服务器端进行检测，文件出现问题及时通知发送方重新发送，避免了接收方收到的损坏文件导致流量的浪费；而一旦文件上传成功而接收方由于网络原因接受失败，则可直接在接收端重新到HTTP服务器进行下载，避免了发送方反复发送相同文件浪费流量。
3. 工程中对一和三方法进行了实现，

方法一：

// 得到文件输入流

FileInputStream inputStream = new FileInputStream(file);

// 转化成字节数组

byte[] buffer = new byte[1024];

int len = 0;

ByteArrayOutputStream outputStream = new ByteArrayOutputStream();

while ((len = inputStream.read(buffer)) > 0) {

outputStream.write(buffer, 0, len);

}

// Base64编码

String result = Base64.encodeBytes(outputStream.toByteArray());

将传输的文件转换成Base64编码字符串，再进行发送。

方法三：

通过HTTP服务器暂存文件时间文件发送

Message信息包格式如下：

<message id="" to="" type="chat">

<subject>file-transfer</subject> //文件传输的标记

<subject xml:lang="zh">file-transfer</subject>

<fileinfo>

<fileName>IMG20150520002516.jpg</fileName> //文件名

<saveFileName></saveFileName> //文件储存名，唯一标示符

<mime\_type>image/jpg</mime\_type> //文件类型

<status>sending</status> //文件发送状态

<date>1432135551162</date> //文件发送的毫秒时间

<md5>25c66b3e7bf676edc47e6905018cd299</md5>//文件的md5校验码

<path>/storage/sdcard0/DCIM/Camera/IMG20150520002516.jpg</path>//文件在发送方的储存路径

</fileinfo>

</message>

1. 其中的saveFileName采用UUID类获取一个唯一时空标识符加上原本文件名作为文件在服务器的储存文件名，避免了多人同时上传同一文件造成的冲突。
2. 上传到HTTP服务器的文件在服务器通过saveFileName获取的hashCode后八位字节转换成的Integer将文件分配到一个特定文件夹中，这是为了将文件打散，从而尽可能的避免冲突和增快检索文件的速度；而接收方同样也可以通过saveFileName得到目标文件的具体地址，从而下载到正确的文件。这样打散文件既加快服务器效率，也方便了收发双方的文件检索。
3. 嵌入式表情开发

通过将表情图片文件存放在drawable文件夹下，通过识别发送和接收信息中的特定编码，将特定编码转换成表情图片显示出来，即用户以为发送了图片，其实只是发送了一个编码字符串，实现用户聊天的多样化，并简化后台的服务器功能实现。

表情发送面板由聊天界面中的表情发送按钮控制其显示和隐藏。

表情编码由\开头，后面三位数字代表drawable中的一个表情，软件先嵌入51个表情，分成三页，三位数字中的第一个数字表示是页数，后两个数字表示该页中的17个表情中的对应表情，从111到127，该编码字符串由正则表达式（“**[/]{1}[1-3]{1}([1]{1}[1-9]{1}|[2]{1}[0-7]{1})**”）进行匹配识别，匹配成功便把四个编码字符转换成表情显示出来，匹配失败就按照原字符串输出显示。

EditText通过SpannableString实现图文并茂，TextView通过SpannableStringBuilder实现图文并茂。

1. APP整体框架：

该APP采用双进程协同工作，一个后台的IMService独占一个进程，维护和Openfire服务器的连接，所有的服务器连接和数据交互操作都在这个进程（由XmppManager进行操作）中执行；而另一个进程用来控制UI的刷新和交互，所有用户交互都在这个进程中实现。

既然前台后台分处不同进程，进程间交互比较困难，故让XmppManager对aidl文件接口实现，使得后台进程的XmppManager对象能够在UI进程中得到一个实例。通过在XmppManager中实现相应的服务器连接和数据交互的方法，就能够在UI进程中通过这个XmppManager实例实现和服务器的交互操作。

即时信息和联系人的刷新通过在IMService进程中将收到的新消息储存在数据库联系人内容提供者和消息内容提供者，在UI进程中通过内容观察者检测对应内容提供者的数据变化，并把变化的信息在UI中进行刷新，实现后台Service维护服务器信息交互，UI前台用户交互协同作用的框架。

五、联系人的首字母排序：

通过将常用姓氏的中文和拼音进行对应，并把拼音以及非汉字的字符串转换成小写字母生成一个用来进行排序的字符串sort，该字符串的首字母就是联系人按照字母分类的首字母T，首字母相同就按照排序字符串sort进行排序。

1. 用户找回密码功能：

当用户忘记密码需要找回时，可通过注册时提供的邮箱找回密码。通过向服务器提供账户名和邮箱，向服务器发送一个IQ数据包，在Openfire服务器端实现一个找回密码的插件，接收这个IQ数据包，对用户的邮箱进行校验，校验通过就向目标邮箱发送一条密码的验证码，同时通知客户端跳转到设定密码的界面，用户到邮箱查看验证码之后，可通过这个验证码对账户密码进行重设，从而实现找回密码的功能。

由于Openfire内部对密码进行封装和加密，因此要从服务器内部获取密码难度较大，因此考虑在账户注册和登录时后台通过加密向服务器端额外发送一组账户密码，通过服务器添加的插件来维护用户的账户密码，实现对账户密码的找回功能。

用户注册时向服务器传输账户、邮箱、密码关联信息

<iq id=”” from=”” to=”” type=””>

<query xmlns = "urn:xmpp:rayo:findpassword" >

<findpassword type = "" password = "MD5加密" account = "" email = "" username=""/>

</query>

</iq>

用户登录时向服务器提交更新此次的密码

<iq id=”” from=”” to=”” type=””>

<query xmlns = "urn:xmpp:rayo:findpassword" >

<findpassword type = "password" code = "" username = "" password =””/>

</query>

</iq>

用户找回密码通过验证码向服务器请求重设密码的请求

<iq id=”” from=”” to=”” type=””>

<query xmlns = "urn:xmpp:rayo:findpassword" >

<findpassword type = "code" account = "" email = "" username=""/>

</query>

</iq>

type:辨别是上传更新密码还是请求重设密码，password表示上传更新密码，code表示重设密码

username:用户名

account:用户JID帐号

password:用户密码，通过MD5加密，此处加密算法封装成方法，可通过更换方法来变更加密

算法

code:验证码

email:Email邮箱

七、Android4.4的拍照和图库功能：因为Android的Api开始，图库的相关Api以及通过图库内容提供者获取的图片Uri有所不同，因此，软件在第一次启动的欢迎界面对Android系统进行检测，并把Android系统版本存入到共享参数中，在App中关于拍照和图库的实现时对共享参数进行检测，以Android4.4为分界线采取两种处理方案，从而实现功能的系统兼容性。

八、登陆界面实现登录功能前，App应该和服务器没有连接，进行登录操作之后才创建和服务器的XmppConnection，启动IMService后台来维护这个连接。要实现这个过程，需要在登陆界面自行创建一个和Openfire服务器的连接，进行账户登录的验证操作，验证成功之后再启动XmppManager的后台IMService维护创建好的连接，并跳转到联系人界面，从而实现对Xmpp连接的维护。