# 一 引言

时移世易，随着21世纪的到来，人们的生活水平和生活质量得到了大幅度的提高，信息化时代也随着科技的不断发展进入了一个新的水平。从之前的广播到报纸，再从报纸到电视，之后再由电脑到现在人们手中的各种智能化手机，人们获取信息的渠道不断增加，通讯技术也得到了飞速的发展，再也不必使用信件，固定电话，人们可以通过手机、电脑和各种手持设备在有网络覆盖的条件下轻轻松松的和同学、朋友、家人进行语音或者视频聊天。

随着经济的发展和现代化建设的不断完善，互联网也逐渐成为了世界的主导。在数十年间，网络的覆盖变得越来越广泛，功能变得越来越强大，它帮助我们拉近了人与人之间的距离，使得整个地球变成了一个真正的“村落”，让远隔于山万水的人体会到了“身隔千里远，情系一线间”的快感。

而如今手持移动设备端的开发也成为了市场需求的主流。手持移动端设备比传统的PC端设备有着携带方便，通讯即时性更高，操作更简单等优势。同时随着3G、4G无线通信技术的更加成熟，三网融合趋于完善，使得在移动端设备之间、移动端设备与PC端设备之间的视频、语音的通信变为了可能。

本课题针对Android系统的移动设备开发一款基于无线局域网应用的即时聊天软件，目的是实现聊天功能，可以广泛应用于交流、资讯、搜索、娱乐、办公等场合的信息交流平台。

# 二 相关内容介绍

## 2.1实现内容

本软件需要实现在无线局域网的条件下，无需在服务器的支持下，利用Android系统的手机，完成点对点之间的通信。作为一款聊天软件不仅要实现基本的一对一的文字聊天功能、群聊功能。

基于以上几点要求，软件应具有以下几类功能需求。

1. 实现对局域网内在线用户的捕获。由于每次进入局域网内的用户有着不定时，不定量的可能性，所以对局域网内当前用户的捕获是完成文字通讯的前提条件。只有在正确获得当前用户的相应列表信息后，本地端才能通过用户的IP地址进行各种交互方案。
2. 能进行文字聊天功能。文字聊天是聊天软件的基本功能之一，在获取到用户IP地址后，通过文字聊天界面用户输入相应文字信息后，本地软件获取用户输入相关文字信息，转换成数据包完成一对一的聊天功能。之后继续扩展完善多人互动聊天的相关功能。

## 2.2运行环境

Android Studio 是基于 IntelliJ IDEA 而来，且适用于开发 Android 应用的官方集成开发环境 (IDE)。无论你要针对 Android 手机、Wear OS by Google 谷歌、Android TV、Android Auto 还是 Android Things 开发应用，Android Studio 都可以胜任，并且 Android Studio 包含每个开发阶段要用到的所有功能。

Android Studio 主要的特色功能包括智能代码编辑器、布局编辑器、性能分析器、APK分析器、模拟器。

### 2.2.1代码编辑器

Android Studio 的编辑器基于 IntelliJ IDEA 而来，可同时支持 Kotlin，Java和C / C ++ 语言，提供了基本补全、智能补全和语句补全功能，还支持创建自定义代码补全模板。

Android Studio 提供了一个名为 Lint 的代码扫描工具，可帮助你发现并更正代码结构质量的问题，而无需执行应用或编写测试。每次编译你的应用时，Android Studio 都会运行 Lint 来检查你的源文件是否有潜在的错误，以及在正确性、安全性、性能、易用性、无障碍性和国际化方面是否需要优化改进。

### 2.2.2 布局编辑器

在布局编辑器中，你可以通过将界面元素拖动到可视化设计编辑器中（而不是手动编写布局 XML），快速构建布局。设计编辑器支持在不同的 Android 设备和版本上预览布局，并且你可以动态调整布局大小，以确保它能够很好地适应不同的屏幕尺寸。

### 2.2.3 性能分析器

如果应用响应速度慢、动画播放不流畅、卡顿、崩溃或极其耗电，则表示其性能差。要避免出现这些性能问题，可以使用 Android Studio 提供的的剖析工具和基准化分析工具来确定应用对哪方面的资源的利用率低下（例如 CPU、内存、显卡、网络和设备电池）。

* 使用 CPU Profiler 检查 CPU Activity 和函数跟踪；
* 使用 Memory Profiler 查看 Java 堆和内存分配；
* 利用 Network Profiler 检查网络流量；
* 使用 Energy Profiler 检查耗电量。

### 2.2.4 APK分析器

Android Studio 包含一个 APK 分析器，可让你在构建流程完成后立即了解 APK 的组成。使用 APK 分析器可以减少调试应用的 DEX 文件和资源相关问题所用的时间，并且有助于减小 APK 的大小。

* 通过 APK 分析器，你可以完成以下操作：
* 查看 APK 中的文件的绝对大小和相对大小；
* 了解 DEX 文件的组成；
* 快速查看 APK 中的文件（如 AndroidManifest.xml 文件）的最终版本；
* 对两个 APK 进行并排比较。

### 2.2.5 模拟器

Android 模拟器可在你的计算机上模拟 Android 设备，这样你就可以在各种设备上以及各个 Android API 级别测试你的应用，而无需拥有每个物理设备。模拟器几乎可以提供真正的 Android 设备所具备的所有功能。你可以模拟来电和短信、指定设备的位置、模拟不同的网速、模拟旋转及其他硬件传感器等等。模拟器随附了针对各种 Android 手机、平板电脑、Wear OS 和 Android TV 设备的预定义配置。

## 2.3实现功能

### 2.3.1 获取在线用户列表模块

在无线局域网中，用户与无线接入点之间走的是无线链路，因此不存在IP地址的分配问题。当用户一旦进入了接入网络，接入网络的AP就会自动给用户分配一个临时的IP，用户在与网络通信的时候都会使用该IP，直到用户离网,IP自动释放。

在线用户列表的获取是完成通信的前提,当用户进入该无线局域网时用户自动被分配到一个随机的IP地址。在获取到本地IP地址后，软件通过UDP报文对本地局域网内的所有IP地址进行广播，如有其它在线用户则会返回相应的应答消息，本地软件收到应答消息后通过ADD\_USER来往本地的用户列表中增加新的用户信息，完成在线用户列表的刷新。

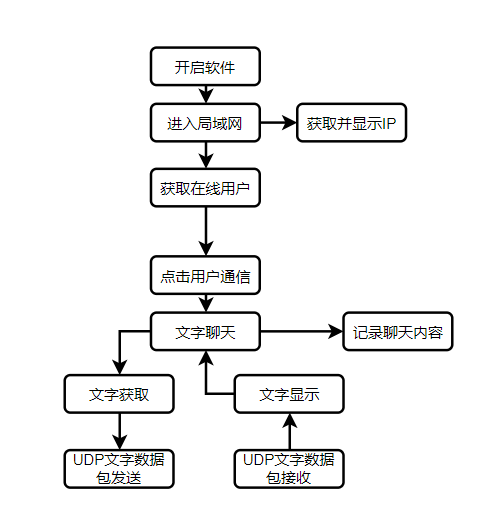
### 2.3.2 文字聊天模块

在用户进入文字聊天界面后，在相应文本输入处输入文字，软件在本地捕获该输入框中的文字后将文字进行打包发送给对方,对方软件收到文字信息的数据包后进行拆包，将文字数据还原并显示在聊天界面中。

### 2.3.3 聊天记录

在聊天记录界面,会记录所有链接IP地址的聊天内容,可通过选择查询对应 IP 用户的聊天信息。另外，该界面除了提供查询的功能，还提供了删除数据的功能，两个删除按钮位于界面上方。

# 三 设计流程图



# 四 详细说明

## 4.1 功能描述

当用户启动聊天活动后，聊天活动进入监听状态，此时向用户展示界面，用户编辑消息并点击发送按钮后，聊天活动启动发送程序，向用户 2 的 RecordFrag 碎片发送消息，该碎片收到消息后，向用户展示消息。

以上仅是用户运行程序时的参考步骤，实际情况比这复杂得多，例如：RecordFrag 碎片中尚未包括 Receiver 的消息接收模块，由于二维时序图展示信息有限，并没有列出该步骤；又如：碎片接收到消息后，并非仅展示消息，还需要对数据进行选择性展示（选择展示在私聊界面还是群聊界面），展示前还需要保存数据进相应的数据表中。

## 4.2 主要功能详述

### 4.2.1 ChatActivity（聊天界面）

在 RecordFragment 中用 Intent 启动 ChatActivity 后，该活动进行初始化，初始化结束后，展示界面如图 7 所示，界面中主要有 6 个控件：返回键、IP 地址显示文本框TextView、选择群聊按钮 ChatInLANBtn、输入框 EditText、发送按钮 SendButton。

（1） 点击返回键可返回启动该活动的上一个活动（图中左上角）；

（2） IP 地址显示文本框：显示当前聊天的用户 IP（图中正上方）；

（3） 选择群聊按钮：选中后发送的所有消息都是以广播的形式在局域网内传播（图中右上角）；

（4） 输入框：发送消息的输入位置（图中左下角）；

（5） 发送按钮：点击后将输入框中的消息取出、发送、删除（图中右下角）。

### 4.2.2 RecordFragment(联系列表)

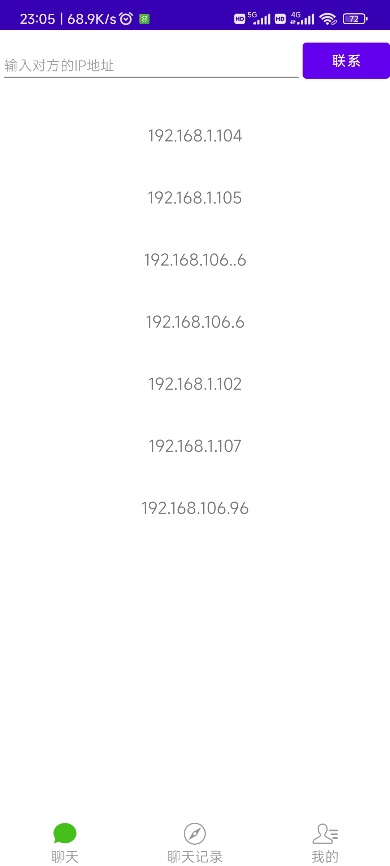
RecordFragment 中展示的是历史信息，主要有三大控件： EditText 文本框、CONTACT 按钮和中间未可视的 RecyclerView。EditText 文本框：可在该框中输入未列举出来的局域网用户 IP 地址，输入对应 IP 后点击 联系即可进入 ChatActivity 界面.

### 4.2.3 HistoryActivity（聊天查询——数据库）

在碎片启动 HistoryActivity 后，该活动开始初始化，除了一些 View 控件进行初始化外，还应扫描数据库，获得所记录的所有 IP 地址，接着初始化 RadioGroup，即初始化图中左侧的所有 IP 对应的 RadioButton，此时控件初始化完毕，可通过选择 RadioButton 查询与对应 IP 用户的聊天信息，此时显示聊天信息于图中靠右的 ListView 中（图中 Item 对应位置即为实际消息显示的位置）。另外，该界面除了提供查询的功能，还提供了删除数据的功能，两个删除按钮位于界面上方。

# 五 结果及分析

## 5.1 结果图



## 5.2 缺陷

本设计实现功能较为简单，在进行深入思考后，在此提出几个缺陷：

1. 本设计通信过程限于局域网， 无法在 Internet 上进行通信。

2. 本设计通信方法采用 UDP 进行通信，当连入 Internet 后，由于网络状态并不稳定，采用该协议无法提供可靠传输。

3. 群聊功能仅限于同一局域网，无法在 Internet 上真正实现群聊。

针对以上缺陷提出的研究方向：

1. 公有 IP 无法在互联网上进行通信，应在云服务器上架构服务端，在客户端以公有 IP访问服务端的私有 IP，即可实现消息转发，从而间接实现“P2P”通信（两客户端通信）。为防止用户过多，更进一步的研究方向：在服务端实现虚拟机，在虚拟机上为用户提供服务，从而实现服务器的便捷扩展，支持更多的用户容量，实现负载均衡。

2. 可采用 TCP 提供 Internet 上的可靠通信，当软件用户量庞大时，可考虑使用 TCP 协议的扩展功能，调整参数，实现更高质量的通信，若还无法满足需求，可考虑借鉴 TCP协议，自定义新的协议规定，使协议更加切合实际网络，进一步提高通信质量。

# 六 课设总结

聊天类app虽然在生活中经常使用,但对于此类的开发并不了解,在此次课设中阅读了一些文献后对于无线通信技术中各类传输协议有了一些了解和认识.对手机端编解码技术有了相应的了解。在前人的研究基础上，本设计针对局域网能够免除流量资费，传输速率高、易于组网等特性，设计并实现了一款在Android平台上能运行的音视频聊天软件。

# 七 主要功能代码

## 7.1 HistoryActivity

public class HistoryActivity extends BasicActivity {  
 private static List<IpMsg> *MsgList*;  
 private static ListView *listView*;  
 private static ArrayAdapter<String> *adapter*;  
 private static Vector<String> *vecString*;  
 private static RadioGroup *ipRadioGroup*;  
 private static String *ipAddress*;  
 private static final String *TAG* = "HistoryActivity";  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*mainfrag\_rcdaty\_his*);  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null) {  
 actionBar.hide();  
 }  
 *ipRadioGroup* = (RadioGroup) findViewById(R.id.*his\_ip\_group*);  
 ((RadioButton)findViewById(R.id.*radio\_for\_size*)).setVisibility(View.*GONE*);  
 *listView* = (ListView) findViewById(R.id.*uav\_info\_list\_view*);  
 *MsgList* = new ArrayList<>();  
 *vecString* = new Vector<>();  
 *adapter* = new ArrayAdapter<String>(HistoryActivity.this,  
 android.R.layout.*simple\_list\_item\_1*, *vecString*);  
 *listView*.setAdapter(*adapter*);  
 *//初始化RadioGroup* initRadioG();  
 *//返回键* ((Button)findViewById(R.id.*his\_call\_back\_btn*)).setOnClickListener  
 (new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 onBackPressed();}  
 });  
 *//删除当前ip数据键* ((Button)findViewById(R.id.*delete\_current\_data\_btn*)).setOnClickListener  
 (new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 int id = *ipRadioGroup*.getCheckedRadioButtonId();  
 if (id == -1) {  
 MyLog.*log*(*TAG*,"尚未选择Ip");  
 Toast.*makeText*(HistoryActivity.this,"请选择Ip",  
 Toast.*LENGTH\_SHORT*).show();  
 }else {  
 final String ipAddr = *ipAddress*;  
 if (!ipAddr.equals("")){  
 DataSupport.*deleteAll* (IpMsg.class,"iPAddress = ?", ipAddr);  
 *vecString*.clear();  
 *adapter*.notifyDataSetChanged();  
 *ipAddress*="";  
 }else {  
 Toast.*makeText*(HistoryActivity.this,"已删除",  
 Toast.*LENGTH\_SHORT*).show();  
 } } } });  
 *//删除所有ip数据键* ((Button)findViewById(R.id.*delete\_all\_data\_btn*)).setOnClickListener  
 (new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 DataSupport.*deleteAll*(IpMsg.class);  
 *vecString*.clear();  
 *adapter*.notifyDataSetChanged();  
 *ipRadioGroup*.clearAnimation();  
 *ipAddress* = "";  
 Toast.*makeText*(HistoryActivity.this,"已删除",  
 Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
 private void initRadioG(){  
 *MsgList* = DataSupport.*select*("ipAddress")  
 .order("ipAddress")  
 .find(IpMsg.class);  
 if (*MsgList*.isEmpty()){  
 MyLog.*log*(*TAG*,"The list is empty when initialize the RadioGroup");  
 }else {  
 String ipInVec = *MsgList*.get(0).getIpAddress();  
 String ipCandidate;  
 Vector<String>ipVec = new Vector<>();  
 ipVec.add(ipInVec);  
 for (IpMsg ipMsg:*MsgList*){  
 MyLog.*log*(*TAG*,ipMsg.getIpAddress());  
 ipCandidate = ipMsg.getIpAddress();  
 if (!ipInVec.equals(ipCandidate)){  
 ipVec.add(ipCandidate);  
 ipInVec = ipCandidate;  
 }  
 }  
 for (final String ipAddr : ipVec){  
 final RadioButton radioButton = new RadioButton(this);  
 final String ipAddress = ipAddr;  
 radioButton.setText(ipAddress);  
 radioButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 HistoryActivity.*ipAddress* = ipAddress;  
 getData(ipAddress);  
 }  
 });  
 *ipRadioGroup*.addView(radioButton);  
 }  
 }  
 }  
 private void getData(String ipAddress) {  
 MyLog.*log*(*TAG*, "in getData,ipAddress:");  
 if (!ipAddress.isEmpty()) {  
 MyLog.*log*(*TAG*, "getData,ipAddress:" + ipAddress);  
 *//查询数据  
 MsgList*.clear();  
 *MsgList* = DataSupport.*where*("iPAddress = ?",ipAddress)  
 .select("time","content","type")  
 .order("time")  
 .find(IpMsg.class);  
 *//更新ListView  
 vecString*.clear();  
 if (!*MsgList*.isEmpty()){  
 for (IpMsg msg : *MsgList*) {  
 String ctn;  
 if (msg.getType()==IpMsg.*TYPE\_RECEIVE*)  
 ctn = msg.getTime()+",Recv:"+""+msg.getContent();  
 else  
 ctn = msg.getTime()+",Send:"+""+msg.getContent();  
 *vecString*.add(ctn);  
 }  
 }else {  
 MyLog.*log*(*TAG*, "数据库为空");  
 Toast.*makeText*(HistoryActivity.this,  
 "数据库为空", Toast.*LENGTH\_SHORT*).show();  
 }  
 *adapter*.notifyDataSetChanged();  
 } else {  
 MyLog.*log*(*TAG*, "ip地址为空");  
 Toast.*makeText*(HistoryActivity.this,  
 "ip地址为空", Toast.*LENGTH\_SHORT*).show();  
 }  
 MyLog.*log*(*TAG*, "out getData");  
 }  
}

## 7.2 MainActivity

public class MainActivity extends BasicActivity {  
 final String TAG = "MainActivity";  
 int currentTabIndex;  
 Fragment[] fragments;  
 ImageView[] imageButtons;  
 TextView[] unreadNumTexts;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null){  
 actionBar.hide();  
 }  
 initTabView();  
 *//设置软键盘不自动弹出* getWindow().setSoftInputMode( WindowManager.LayoutParams.*SOFT\_INPUT\_STATE\_HIDDEN*);  
 }  
 @Override  
 protected void onDestroy(){  
 super.onDestroy();  
 }  
 private void initTabView() {  
 RecordFrag rcFrag = new RecordFrag();  
 FriendFrag frdFrag = new FriendFrag();  
 FunctionFrag funcFrag = new FunctionFrag();  
 HistoryActivity historyActivity=new HistoryActivity();  
 *//CallFrag callFrag = new CallFrag();  
 //fragments = new Fragment[] {rcFrag,frdFrag,funcFrag,callFrag};* fragments = new Fragment[] {rcFrag,frdFrag,funcFrag};  
 *//三个title\_bottom小红点* unreadNumTexts = new TextView[3];  
 unreadNumTexts[0] = (TextView) findViewById(R.id.*title\_rc\_unread\_num*);  
 unreadNumTexts[1] = (TextView) findViewById(R.id.*title\_frd\_unread\_num*);  
 unreadNumTexts[2] = (TextView) findViewById(R.id.*title\_func\_unread\_num*);  
 *//获取title\_bottom四个img* imageButtons = new ImageView[4];  
 imageButtons[0] = (ImageView) findViewById(R.id.*title\_rc\_img*);  
 imageButtons[1] = (ImageView) findViewById(R.id.*title\_frd\_img*);  
 imageButtons[2] = (ImageView) findViewById(R.id.*title\_func\_img*);  
 *//imageButtons[3] = (ImageView) findViewById(R.id.title\_call\_img);* imageButtons[0].setSelected(true);  
 *// 添加显示fragment* getSupportFragmentManager().beginTransaction()  
 .add(R.id.*main\_frameLayout*, rcFrag)  
 .add(R.id.*main\_frameLayout*, frdFrag)  
 .add(R.id.*main\_frameLayout*, funcFrag)  
 *//.add(R.id.main\_frameLayout, callFrag)* .hide(frdFrag).hide(funcFrag)  
 *//.hide(callFrag)* .show(rcFrag).commit();  
 currentTabIndex = 0;  
 }  
 public void onTabChange(View v){  
 int index = Integer.*MAX\_VALUE*;  
 switch (v.getId()){  
 case R.id.*title\_rc\_btn*:  
 if(currentTabIndex != 0){  
 index = 0;  
 }  
 break;  
 case R.id.*title\_frd\_btn*:  
 if(currentTabIndex != 1){  
 index = 1;  
 }  
 break;  
 case R.id.*title\_func\_btn*:  
 if(currentTabIndex != 2){  
 index = 2;  
 }  
 break;  
*// case R.id.title\_call\_btn:  
// if(currentTabIndex != 3){  
// index = 3;  
// }  
// break;* }  
 if (index != Integer.*MAX\_VALUE*){  
 if(!fragments[index].isAdded()){  
 getSupportFragmentManager().beginTransaction().add  
 (R.id.*main\_frameLayout*, fragments[currentTabIndex]);  
 }  
 FragmentTransaction trx = getSupportFragmentManager().beginTransaction();  
 if(index != currentTabIndex){  
 trx.hide(fragments[currentTabIndex]);  
 trx.show(fragments[index]).commit();  
 imageButtons[currentTabIndex].setSelected(false);  
 imageButtons[index].setSelected(true);  
 }  
 currentTabIndex = index;  
 }  
 }  
}

## 7.3 ChatActivity

public class ChatActivity extends BasicActivity {  
 public List<IpMsg> msgList;  
 private static EditText *inputText*;  
 private static RecyclerView *msgRecyclerView*;  
 private static TextView *ipView*;  
 private static RadioButton *chatInLANrBtn*;  
 public static MsgAdapter *adapter*;  
 public static String *ipAddr*;  
 private static String *ipAddrForLAN*;  
 private static boolean *isRadioCkd*;  
 private static IntentFilter *intentFilter*;  
 private static LocalReceiver *localReceiver*;  
 public String inputCtn;  
 private static final String *TAG* = "ChatActivity";  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*mainfrag\_rcdaty\_chat*);  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null){  
 actionBar.hide();  
 }  
 */\*\*\*初始化\*\*\*/  
 ipView* = (TextView)findViewById(R.id.*chat\_ip\_view*);  
 *inputText* = (EditText)findViewById(R.id.*input\_text*);  
 *chatInLANrBtn* = (RadioButton)findViewById(R.id.*chat\_LAN\_radio*);  
 *msgRecyclerView* = (RecyclerView)findViewById(R.id.*msg\_recycler\_view*);  
  
 *isRadioCkd* = false;  
 msgList = new ArrayList<>();  
 initMsgs(); *//查询数据库，初始化msgList* LinearLayoutManager layoutManager = new LinearLayoutManager(this);  
 *msgRecyclerView*.setLayoutManager(layoutManager);  
 *adapter* = new MsgAdapter(msgList);  
 *msgRecyclerView*.setAdapter(*adapter*);  
 *msgRecyclerView*.scrollToPosition(msgList.size()-1); *//定位到记录的最后一行  
 intentFilter* = new IntentFilter();  
 *localReceiver* = new LocalReceiver();  
 *intentFilter*.addAction("com.enzosalvetore.circle.LOCAL\_BROADCAST");  
 RecordFrag.*localBroadcastManager*.registerReceiver  
 (*localReceiver*,*intentFilter*);  
  
 ((Button)findViewById(R.id.*chat\_call\_back\_btn*)).setOnClickListener  
 (new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 onBackPressed();  
 }  
 });  
  
 *chatInLANrBtn*.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 if (*isRadioCkd*){  
 MyLog.*log*(*TAG*,"点击Radio前已选中");  
 *chatInLANrBtn*.setChecked(false);  
 *ipAddr* = *ipAddrForLAN*; *//还原原通信ip* }else {  
 MyLog.*log*(*TAG*,"点击Radio前没有选中");  
 *ipAddrForLAN* = *ipAddr*; *//保留当前通信ip  
 chatInLANrBtn*.setChecked(true);  
 *ipAddr* = "255.255.255.255";  
 }  
 *isRadioCkd* = !*isRadioCkd*;  
 }  
 });  
  
 ((Button)findViewById(R.id.*send\_btn*)).setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 final String content = *inputText*.getText().toString();  
 if(!"".equals(content)){  
 final IpMsg msg = new IpMsg(content,IpMsg.*TYPE\_SENT*);  
 msgList.add(msg);  
 *//有新消息时刷新RecyclerView的显示  
 adapter*.notifyItemInserted(msgList.size()-1);  
 *//将RecyclerView定位到最后一行  
 msgRecyclerView*.scrollToPosition(msgList.size()-1);  
 inputCtn = *inputText*.getText().toString();  
 new Thread(new Runnable() {  
 @Override  
 public void run() {  
 *sendByUDP*(*ipAddr*,inputCtn);  
 RecordFrag.*saveData*(*ipView*.getText().toString(),  
 content,IpMsg.*TYPE\_SENT*);  
 }  
 }).start();  
 *inputText*.setText("");  
 }  
 }  
 });  
 }  
  
 @Override  
 protected void onDestroy(){  
 *ipAddr* = "";  
 RecordFrag.*localBroadcastManager*.unregisterReceiver(*localReceiver*);  
 super.onDestroy();  
 }  
  
 private void getData(String ipAddress) {  
 MyLog.*log*(*TAG*, "in getData");  
 if (!ipAddress.isEmpty()) {  
 MyLog.*log*(*TAG*, "getData,ipAddress:" + ipAddress);  
 msgList.clear();  
 msgList = DataSupport.*where*("iPAddress = ?",ipAddress)  
 .select("time","content","type")  
 .order("time")  
 .find(IpMsg.class);  
 if (msgList.isEmpty()){  
 MyLog.*log*(*TAG*, "数据库为空");  
 Toast.*makeText*(ChatActivity.this,  
 "数据库为空", Toast.*LENGTH\_SHORT*).show();  
 }else {  
 MyLog.*log*(*TAG*,"查询数据库结果：");  
 for (IpMsg ipMsg:msgList){  
 MyLog.*log*(*TAG*,ipMsg.getTime()+"ip:"+  
 ipMsg.getIpAddress()+":"+ipMsg.getContent());  
 }  
 }  
 } else {  
 MyLog.*log*(*TAG*, "ip地址为空");  
 Toast.*makeText*(ChatActivity.this,  
 "ip地址为空", Toast.*LENGTH\_SHORT*).show();  
 }  
 MyLog.*log*(*TAG*, "out getData");  
 }  
  
 private void initMsgs(){  
 Intent intent = getIntent();  
 *ipAddr* = intent.getStringExtra("ipAddress");  
 if ("".equals(*ipAddr*)) onBackPressed();  
 *ipView*.setText(*ipAddr*);  
 getData(*ipAddr*); *//初始化List* }  
  
 public static void sendByUDP(String ipAddr,String inputCtn){  
 if (!inputCtn.equals("")){  
 try{  
 DatagramSocket ds = new DatagramSocket();  
 byte buf[] = inputCtn.getBytes();  
 int size = buf.length;  
 MyLog.*log*(*TAG*,ipAddr+","+RecordFrag.*port*);  
 MyLog.*log*(*TAG*,"Send>>content:"+inputCtn+"size:"+size);  
 DatagramPacket dp = new DatagramPacket(buf,size,  
 InetAddress.*getByName*(ipAddr),RecordFrag.*port*);  
 ds.send(dp);  
 MyLog.*log*(*TAG*,"已发送");  
 ds.close();  
 }catch (IOException e){  
 e.printStackTrace();  
 }  
 }  
 }  
  
 class LocalReceiver extends BroadcastReceiver{  
 @Override  
 public void onReceive(Context context,Intent intent){  
 final String ctn = intent.getStringExtra("recv\_msg");  
 if (!ctn.equals("")){  
 final IpMsg ipMsg = new IpMsg(ctn,IpMsg.*TYPE\_RECEIVE*);  
 MyLog.*log*(*TAG*,"收到广播信息:"+ipMsg.getContent());  
 msgList.add(ipMsg);  
 *adapter*.notifyItemInserted(msgList.size()-1);  
 *msgRecyclerView*.scrollToPosition(msgList.size()-1);  
 }  
 }  
 }  
}

## 7.5 AndroidManifest

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.ymx.testproject1">  
 *<!-- 允许应用程序改变网络状态 -->* <uses-permission android:name="android.permission.CHANGE\_NETWORK\_STATE" />  
 *<!-- 允许应用程序改变 WIFI 连接状态 -->* <uses-permission android:name="android.permission.CHANGE\_WIFI\_STATE" />  
 *<!-- 允许应用程序访问有关的网络信息 -->* <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
 *<!-- 允许应用程序访问 WIFI 网卡的网络信息 -->* <uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />  
 *<!-- 允许应用程序完全使用网络 -->* <uses-permission android:name="android.permission.INTERNET" />  
 <uses-permission android:name="android.permission.READ\_PHONE\_STATE" />  
 <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
 <application  
 android:name="org.litepal.LitePalApplication"  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.Testproject1">  
 <activity  
 android:name=".MainActivity"  
 android:screenOrientation="portrait"  
 android:launchMode="singleTask"  
 android:exported="true">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity><activity android:name=".mainFragment.rcdAty.ChatActivity"  
 android:screenOrientation="portrait">  
 </activity>  
 <activity android:name=".mainFragment.rcdAty.HistoryActivity"  
 android:screenOrientation="portrait">  
 </activity>  
 </application>  
</manifest>