局域网搜索设置APK要求

分手机端和设备端

手机端：

1. 手机端搜索局域网内设备;
2. 搜索到设备并通过验证后，能获取设备端系统时间，语音，ETHERNET IP 地址和，音量和联系人列表。
3. 对以上信息修改后能发回设备，更新设备端设置和联系人。
4. 在APK打开期间发送KEEP ALIVE 包。使设备端保持端口打开状态。
5. UI分三个页面：
   1. 搜索设备，显示并修改设备端系统时间，语音，ETHERNET IP 地址和，音量;
   2. 显示或更新设备端联系人列表;
   3. 修改联系人内容。

设备端：

1. 在设备上按下F10键后开始监听端口（8888），如果检测到有手机端向此端口发送信息。则验证手机端身份。在10分钟内没收到信息则停止监听。在已经与一个手机端在连接状态下，收到另一个手机端的连接请求，回复正忙。
2. 身份验证后可以接受手机端请求发送系统时间，语音，ETHERNET IP 地址和，音量和联系人列表，或接受手机的更新数据。
3. 在20分钟内没有接收到手机端的新的数据或KEEP ALIVE 包，关闭8888端口。
4. 没有UI, 为方便调试可以有一个页面设置连接状态。

UdpBroadcastSearcher —— UDP广播搜索者

用于局域网中，在不知道设备的内网IP的情况下，通过UDP广播搜索局域网内所有的的设备

一、导入

方法1：直接导入源码

把工程下的searcher文件夹里的源码全部导入即可

方法2：导入jar包

在工程下有一个searcher.jar包，拷贝到自己的工程中

二、使用方法

2.1 不携带用户数据的使用（最简单的使用）

只要实现内部的抽象方法即可。

主机：

new SearcherHost() {

@Override

public void onSearchStart() {

...

}

@Override

public void onSearchFinish(Set deviceSet) {

//所有设备集合都在deviceSet中，通过getIp()可获取设备IP，getPort()获取端口

...

}

@Override

public void printLog(String log) {

...

}

}.search();

设备：

// 创建设备SearcherDevice mSearcherDevice = new SearcherDevice() {

@Override

public void onDeviceSearched(InetSocketAddress inetSocketAddress) {

// 主机IP: inetSocketAddress.getAddress().getHostAddress()

// 主机端口：inetSocketAddress.getPort()

...

}

@Override

public void printLog(String s) {

...

}

};

// 打开设备

mSearcherDevice.open();

// 关闭设备

mSearcherDevice.close();

demo参考工程下的AppHost和AppDevice

2.2 携带用户数据的使用

若需要携带用户数据，必须重写类的用户数据打包与解包方法。

主机：

【可选】在构造方法中，指定泛型的具体类型。若不指定，parseUserData()也许需要强转。

在构造方法中，设置用于接收设备用户数据的最大字节长度（≥设备发送的用户数据长度）

在构造方法中，设置设备类的字节码，没有重写，则写SearcherHost.DeviceBean.class（用于内部通过反射创建实例对象）

重写两个用户数据打包方法： packUserData\_Search()和packUserData\_Check()

重写用户数据解析方法： parseUserData(byte type, MyDevice device, byte[] userData)

SearcherHost searcherHost = new SearcherHost<MyDevice>(1024, MyDevice.class) {

@Override

public void onSearchStart() {

...

}

@Override

public void onSearchFinish(Set deviceSet) {

...

}

@Override

public void printLog(String log) {

...

}

/\*\* \* 重写以下3个方法 \*/

/\*\* \* 打包搜索时数据 \*/

@Override

protected byte[] packUserData\_Search() {

...

}

/\*\* \* 打包核对时的数据 \*/

@Override

protected byte[] packUserData\_Check() {

...

}

/\*\* \* 解析用户数据 \* @param type 类型。搜索申请or搜索确认 \* @param device 设备 \* @param userData 数据 \* \* @return true-解析成功 \*/

@Override

public boolean parseUserData(byte type, MyDevice device, byte[] userData) {

...

}

};

// 开始搜索

searcherHost.search();

设备：

在构造方法中，设置用于接收设备用户数据的最大字节长度（≥主机发送的用户数据长度）

重写设备响应时的用户数据打包方法： packUserData()

重写用户数据解析方法： parseUserData(byte type, byte[] userData)

【可选】重写判断是否为本机ip地址的方法（主要用于判断主机确认返回的ip地址是否正确，默认返回true）： isOwnIp(String ip)

SearcherDevice searcherDevice = new SearcherDevice(1024) {

@Override

public void onDeviceSearched(InetSocketAddress inetSocketAddress) {

...

}

@Override

public void printLog(String s) {

...

}

/\*\* \* 重写以下2个方法 \*/

/\*\* \* 响应时的打包数据 \*/

@Override

protected byte[] packUserData() {

...

}

/\*\* \* 解析用户数据 \* @param type 类型，搜索请求or搜索确认 \* @param userData 用户数据 \* @return 解析结果是否成功 \*/

@Override

public boolean parseUserData(byte type, byte[] userData) {

...

}

/\*\* \* 判断ip是否是本机ip \* @param ip 判断的ip地址 \* @return \*/

@Override

public boolean isOwnIp(String ip) {

...

}

};

searcherDevice.open();

demo参考工程下的se\_host和se\_device