## 绪论

### 课题的背景及意义

### 研究现状

## 蓝牙协议

### 蓝牙协议体系

蓝牙协议中包含一个协议体系，是众多协议的一个总称，我们按照SIG的关注程度可以把它分成四层：第一层是核心协议，包括BaseBand、LMP、L2CAP、SDP；第二层是电缆替代协议RFCOMM,第三层是电话传送控制协议，有TCS-Binary、AT命令集；第四层是选用协议，主要有PPP、UDP/TCP/IP、OBEX、WAP等协议。

蓝牙核心协议

基带和链路控制层确保网络中的每个蓝牙设备单元之间的物理连接用的是射频。蓝牙无线射频系统是一种跳频系统，该系统是在指定的时隙、指定频率和跳频系统中发送的。它采用查询和寻呼过程同步不同设备之间发送和时钟，提供了两种物理连接方法，就是面向连接（SCO）和无连接（ACL），另外，在同一广播可以实现多路数据传输。ACL适用于分组数据，组织对语音和语音与数据相结合，所有的语音和数据包都伴随着不同程度的在纠错（FEC）或循环冗余校验（CRC），并且可以加密。此外，为不同的数据类型（包括连接管理信息和控制信息）被分配一个特殊的通道。

### Android中的蓝牙协议

安卓平台包括支持蓝牙网络协议栈，它允许设备与其他蓝牙设备进行无线交换数据。应用程序框架提供通过安卓蓝牙接口访问蓝牙功能。这些应用程序的无线连接到其他蓝牙设备，使点对点和多点的无线功能。

使用蓝牙接口，安卓应用程序可以执行以下：

其他蓝牙设备的扫描

查询本地蓝牙适配器，用于配对蓝牙设备

建立RFCOMM通道

通过服务发现连接到其他设备

从其他设备传输数据

管理多个连接

本文档介绍了如何使用经典蓝牙。经典蓝牙是更为正确的选择，更多的电池密集的操作，如流媒体和通信之间的安卓设备。对于低功耗的蓝牙设备，安卓4.3（原料药18）对蓝牙低能量的支持。要了解更多，见蓝牙低能量。

本文档介绍了如何使用安卓的蓝牙接口来完成四个主要任务所需的通信使用蓝牙：设置蓝牙，寻找设备，无论是配对或可在本地区，连接设备，设备之间传输数据。

所有的蓝牙API可在android.bluetooth包。这里的类和接口，您将需要创建蓝牙连接的总结：

蓝牙适配器

表示本地蓝牙适配器（蓝牙收音机）。对于任务是所有蓝牙互动的切入点。利用这一点，你会发现其他的蓝牙设备，查询列表（配对）保税设备，实例化一个蓝牙设备使用一个已知的MAC地址，并创建一个bluetoothserversocket听来自其他设备的通信。

蓝牙设备

表示远程蓝牙设备。使用此请求与远程设备的连接通过一个BluetoothSocket或查询有关的装置，如名称、地址、类别信息，并结合状态。

BluetoothSocket

代表一个蓝牙套接字接口（类似于TCP套接字）。这是连接点，允许一个应用程序与另一个蓝牙设备通过InputStream和OutputStream交换数据。

bluetoothserversocket

代表一个打开服务器套接字侦听传入的请求（类似于一个TCP ServerSocket）。为了连接2个安卓设备，一个设备必须打开一个与这个类的服务器套接字。当一个远程蓝牙设备进行连接请求的这种装置的bluetoothserversocket将返回一个BluetoothSocket连接时连接被接受。

蓝牙类

描述蓝牙设备的一般特性和功能。这是一组只读的属性，定义设备的主要和次要设备类及其服务。然而，这并不能可靠地描述设备所支持的所有蓝牙配置文件和服务，但对设备类型的提示是有用的。

bluetoothprofile

表示蓝牙配置文件的接口。蓝牙配置文件是一种基于蓝牙的设备之间的无线接口规范。一个例子是双手自由配置文件。对于配置文件的更多讨论，请参见使用配置文件

bluetoothheadset

提供了对蓝牙耳机的支持。这包括蓝牙耳机和免提（V1.5）简介。

bluetootha2dp

定义如何高质量的音频可以从一个设备传输到另一个蓝牙连接。”A2DP”代表先进的音频分布。

bluetoothhealth

表示控制蓝牙服务的健康设备配置文件代理。

bluetoothhealthcallback

一个抽象类，您使用来实现bluetoothhealth回调。您必须扩展这个类并实现回调方法以接收应用程序注册状态和蓝牙信道状态的变化的更新。

bluetoothhealthappconfiguration

表示蓝牙健康第三方应用程序注册与远程蓝牙健康设备通信的应用程序配置。

bluetoothprofile.servicelistener

一个界面，通知bluetoothprofile IPC客户当他们已连接或断开了与服务（即内部服务运行一个特定的配置文件）。

## Android开发技术

### 系统历史

Android是基于linux开发的供移动设备使用的操作系统，它是开源的。Android系统最初是由个人开发的，他就是andi，刚开始时主要用来给数码相机使用，但是效果却不理想。而恰好在这个时候智能手机开始快速发展，于是android系统被改造成了一款用于智能手机的操作系统了。后来，这个系统被美国的科技公司google收购，逐渐走上正轨。

### 主要特征

Android系统是主要工作在智能手机之上的基于linux的操作系统，处处体现了与手机相关的特性。因为硬件的区别，android系统与window操作系统的区别特别大，只有充分了解Android系统特性，才能成功的开发出适合手机运用的app。以下列出来的是Android SDK4.0的主要特性：

（1） 应用库层已经准备好各种开发控件，开发人员也可以自定义。

（2）专门为android系统设计的Dalvik虚拟机，在内存优化垃圾管理方面有着更优的效率。

（3）集成了基于开源Webkit引擎的网页浏览器。

（4）经过优化过的自定义2D图形库和基于OpenGL ES1.0库可选硬件加速。

（5）基于sql的封装过的Sqlite数据库，使用更加方便。

（6）支持常规的音频视频多媒体格式，如MPEG4、H.264、MP3、AAC、AMR、JPG、PNG、GIF GSM电话(硬件决定)

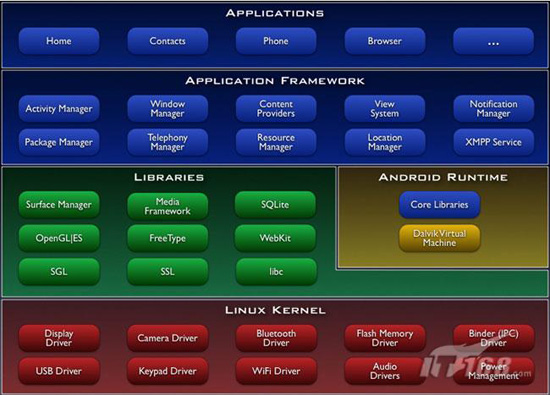
（7）支持蓝牙开发、GPRS-EDGE、3G和Wifi无线网络

（8）支持相机、导航、GPS全球定位、和感应加速器

（9）丰富的开发环境包含设备模拟器（不用再真机上面调试）、内存工查看工具、调试器、内存和性能分析工具，以及Eclipse和android studio集成开发环境插件

### 系统架构

系统架构如图所示：



Android系统分成四层，从下到上依次是linux内核层，系统运行库层，应用程序框架层，应用层。

Linux内核层：android系统是基于linux系统的，所以内核就是linux，它的核心服务基于linux2.6内核，处理的主要任务包括内存管理，进程调度，网络协议栈和驱动模型以及安全性处理等。

系统运行库层：系统运行库包括程序库和android运行库，程序库就包含了一些c和c++库，因为有些功能用c来实现效率会更高，系统c库是继承自标准的c系统函数库，主要用来给embedded linux服务，而基于packetVideo openCore的媒体库主要用来各种音频视频格式的文件的播放和录制，编码的格式有MP4，mp3等等。Surface Manager是用来显示子系统的管理，为各种应用程序提供3d和2d图层的融合。该程序库中还包含底层的2d图形引擎SGL，基于位图和矢量的字体显示引擎FreeType，以及提供轻量级数据库的sqlite。

应用程序框架层：应用程序框架层就是我们程序员所熟知的framework层，它为应用程序提供了用于访问核心内核的api框架，这样的一种设计简化了程序员的操作，避免了重复早轮子，极大地提高了android程序员的开发效率。这一层的内容包括各种view，内容提供器，资源管理器，通知管理器，活动管理器等等。各种view可以用来设计应用程序的界面，各种各样的app界面就是由这些view搭配构建而成，当然我们也可以给予framework层创建自己需要的view，这就称为自定义view。资源管理器顾名思义就是管理各种各样的资源，比如应用程序中的图片，需要展示的文字等等，有了资源管理器，我们在使用我们的资源的时候就会更加的方便。活动管理器就是就是管理activity的管理器，负责activity的创建关闭。

应用程序层：这就是我们所使用的应用程序了，包括系统的联系人，拨号器，浏览器等等。

### Activity的介绍

活动是一个应用程序组件，它提供了一个屏幕，用户可以进行交互，以便做一些事情，如拨打电话，拍照，发送电子邮件，或查看地图。每个活动都有一个窗口来绘制用户界面。窗口通常填充在屏幕上，但可能比屏幕小，浮动在顶部的其他窗口。

一个应用程序通常由多个松散绑定到彼此的活动组成。通常情况下，在应用程序中的一个活动被指定为“主”活动，该活动在第一次启动应用程序时呈现给用户。每一个活动都可以启动另一个活动来执行不同的动作。每次一个新的活动开始，以前的活动被停止，但系统保留在堆栈中的活动（“回栈”）。当一个新的活动开始时，它被推到后面堆栈，并以用户为焦点。回堆栈遵守基本的“后进先出栈”机制，因此，当用户完成当前活动并按下返回按钮，则弹出堆栈（和破坏）和恢复前一个活动。（在任务和反堆栈文件中，后面的堆栈被讨论。）

##### Activity的生命周期

Activity的生命周期如图所示



具体的生命周期的调用情况如下：

1、开始活动：系统将调用onCreate方法，接着调用OnStart方法，最后调用onResume进入运行状态的活动。这种情况是最常见的，活动的启动都是这个流程。

2、当前活动覆盖或锁定屏幕上的其他活动：系统会调用onPause方法，当前活动的暂停执行，也就是当前活动失去了焦点，不能再获取输入或者点击事件。

3、目前的活动由没有焦点到获取焦点，系统会调用onResume方法，获取到焦点再次进入运行状态，注意在onResume方法调用的前后活动一直都是可见的。

4、在活动去新的活动界面或按Home键回到主屏幕背景：系统将先调用onPause方法，然后调用onStop方法进入停滞状态，此时有两个变化，首先是失去焦点即不能获取输入输入事件和点击事件，然后由可见状态变成不可见。

5、当用户回到这个活动，系统会调用onrestart方法，然后调用OnStart方法，最后调用onResume方法，再次进入运行状态，这是一个比较特殊的情况，也是onrestart会被调用的情况。

6、用户退出当前活动：系统首先调用onPause方法，然后调用onStop方法，最后系统会调用ondestory方法，当前活动结束，此时活动不仅不可见，而且活动是已经被销毁了，如果要重新使用这个活动需要调用oncreate方法了。

Onresume和onPause方法控制的是活动是否获取焦点，而onStart方法和onStop方法控制的是活动是否可见，这两组方法之间是由区别的。

##### Activity的四种启动模式：

Standard：标准启动模式，每次使用都新创建一个活动，不再复用之前的活动。  
singleTop：如果需要的活动在栈顶，便复用这一个活动，不再新创建。  
singleTask：如果栈中存在之前的活动，则复用这一个活动，不再新创建，而且把栈中在这个活动之上的其他活动全部清除掉。  
singleInstance：只创建一个活动，一直复用。

### Service

一个服务是一个组件，运行在后台执行长时间运行的操作或执行远程进程的工作。服务不提供用户界面。例如，一个服务可能在后台播放音乐，而用户在不同的应用程序中，或者它可以在网络上获取数据，而不阻塞用户交互的活动。另一个组件，例如一个活动，可以启动服务，让它运行或绑定到它，以便与它进行交互。

服务不是一个单独的过程。服务对象本身并不意味着它正在运行在它自己的进程中；除非另有说明，它在同一个进程中运行，作为应用程序的一部分。服务不是线程。它不是一个意思本身做工作的主要线程（以避免应用程序没有响应错误）。因此，服务本身实际上是非常简单的，提供了2个主要功能：应用程序的一个设施，告诉系统关于它想在背景中做的事情（即使用户没有直接与应用程序交互）。这相当于调用上下文，startservice()后服务将一直在后台运行，直到被系统关闭。

Service是运行在主线程中的，我们不能直接在service中做一些复杂的。用适当的行为来实现这些服务，比如创建一个二级线程，在它完成它的工作。

注意，因为服务本身就是这么简单，你可以让你对它的简单或复杂的：你想从把它作为一个地方的java对象，你直接调用方法（由当地服务的示例如图所示），提供一个使用AIDL全可进行远程操作的接口。

服务的生命周期

有两点原因，一个服务可以由系统运行。如果有人打电话的背景。startservice()然后系统会检索服务（创造它，如果需要调用它的oncreate()方法）然后调用onStartCommand（其意图，int，int）由客户提供的参数的方法。该服务将在这一点上继续运行，直到stopservice()或stopself()称为上下文。请注意，多次调用上下文。startservice()不窝（尽管他们在onstartcommand()对应的多个电话做的结果），所以无论多少次，它开始一个服务将停止一次背景。stopservice()或stopself()叫做；然而，服务可以用stopSelf（int）来保证服务的方法不停到开始的意图已被处理。

为启动的服务，有两个主要的操作模式，他们可以决定运行，取决于他们的回报onstartcommand()价值：start\_sticky用于显式启动和停止所需要的服务，而start\_not\_sticky或start\_redeliver\_intent用于服务，应运行在处理任何命令发送给他们。在语义上查看更多细节的链接文档。

客户也可以使用的上下文。bindservice()获得持久连接到服务。这同样造成服务如果它没有运行（虽然这样叫oncreate()），但不要求onstartcommand()。客户端将收到的IBinder对象，从onBind服务回报（意图）的方法，让客户再打电话回服务。该服务将运行只要建立连接（客户端是否仍保留在服务的IBinder参考）。通常返回的是一个复杂的界面的时候，已经写在AIDL。