基于 Android 的网络更新功能的研究与实现

陈剑，张小频

（北京邮电大学信息光子学与光通信研究院国家重点实验室，北京 100876）

摘要：自从 Android 系统推出以来，凭借 Android 自身的特性，Android 已经成为移动智能手机的主流平台。伴随着互联网技术的不断发展，基于 Android 平台的移动互联网应用也得到了大力的发展。在本文中，主要研究的是在 Android 平台中对应用进行网络更新，并实现断点续传下载。采用 C/S 客户端-服务器模式，基于 HTTP 协议进行通信。在 Android 应用上实现版本的更新检测，使用 Service 组件和多线程技术从服务器端下载最新的 apk，并广播通知用户安装。断点续传功能采用 SQLite 数据库进行实现。

关键词：Android；C/S 模式；断点续传；多线程技术

中图分类号：TP311.1

Research and implementation of network update function based on Android

Chen Jian, Zhang Xiaopin

(State Key Laboratory of Information Photonics and Optical Communication,Beijing University of Posts and Telecommunications,Beijing 100876)

Abstract: Since the launch of Android system,Android has become the mainstream platform on the mobile smart phones with its own characteristics. With the continuous development of internet technology,mobile internet applications based on the Android platform have also been strong growth.In this paper,the main research is how to update the application which based on Android platform through the network,and impletment resume broken transfer.Client-server model is used for the design, the communication is based on the HTTP protocol. Implements the update detection of the application version on Android application, the Service component and multi-threading technology is used to download the latest apk from the server,and the Broadcast Receiver is used to notify the user to install.The SQLite database is used to implement the resume broken transfer.

Key words: Android; C/S model; Resume broken transfer; Multi-threading technology

0 引言

自从互联网巨头 Google 公司推出 Android 系统以来，凭借着 Android 自身代码开源、使用便捷、灵活等诸多特点在 3G 和无线网络领域持续走红，得到了越来越多的用户和设备商的青睐。根据 IDC 的最新数据显示，2013 年第一季度，全球 Android 智能手机出货量上升至 1.621 亿台，高于去年同期，且市场份额为 75%。数据表明：Android 平台已经是当前最为流行的平台。虽然苹果的 iPhone 和微软的 Windows Phone 等也在快速发展，但仍然无法撼动 Android 在市场上的领先地位。国家正在大规模建设 3G 网络以及 4G 网络即将到来，移动互联网真正意义上进入了一个高速发展的阶段。中国的移动互联网用户人数将在 2014 年 5 月左右增至 4 亿[1]。移动与互联网的结合成为了必然趋势，这给智能手机终端带来了广阔的发展空间。而移动应用总是在不断的改进，推出新的版本。因此，在应用中增加应用网络更新功能是非常必要的，也是未来应用发展的趋势。

在本文中，主要的研究主题是在基于 Android 平台的应用中，实现应用的网络更新。首

作者简介：陈剑（1988-），男，硕士研究生，主要研究方向：移动通信与计算机应用

通信联系人：张小频（1957-），男，教授，硕士生导师，主要研究方向：光纤通信、无线通信、移动通信的研究，E-mail：xiaopinzhang@bupt.edu.cn. E-mail: xiaopinzhang@bupt.edu.cn

先简要介绍了 Android 平台及其相关组件，接着介绍了网络更新开发中用到的关键技术，包括 xml 解析技术、多线程技术，以及 TOMCAT 服务器。然后详细阐述了客户端-服务器端的通信原理，以及网络更新功能的设计与实现。实现主要分为 3 部分：版本检测、应用下载、广播通知用户下载完成，以及断点续传功能的实现。

1 Android 平台简介

Android 的中文意思是“机器人”，本身是一个操作系统，是基于 Linux 开放性内核的。Android 作为一个移动设备的平台，其软件层次结构包括了 4 层，自下而上分别是：Linux 内核层、本机库（Libraries）和 Android 运行环境（RunTime）、应用程序框架（Application FrameWork）、应用程序（Application）[2]。

此外，Android 应用架构是基于组件的，这些组件的通信在 AndroidManifest 文件中通过描述的 Intent 来实现。下面简单介绍 Android 的四大组件。

（1）Activity：一个 Activity 就是一个单独的屏幕。Activity 是应用程序的入口，为用户提供了进行交互的界面窗口，在 Activity 上可以监听并处理用户的事件并作出相应的响应。

（2）Service：Service 没有提供与用户进行交互的表示层，Service 是在一段不确定的时间运行在后台的应用组件。当应用程序需要进行某种不需要前台显示的计算或数据处理时，就可以启动一个 Service 来完成[3]。

启动 Service 有两种方式，通过 startService 方法或 bindService 方法，二者的生命周期不尽相同。当系统调用 startService 方法时，系统依次调用 onCreat()方法和 onStart()方法来启动。当调用 stopService()、Service 调用自身的 stopSelf()时，Service 才会停止执行。通过 bindService 方法启动时，调用 onCreate()方法完成初始化工作，然后将此 Service 和 Context 对象（如 Activity）进行绑定，当被绑定的 Context 对象被销毁时，与之绑定的 Service 也会停止运行。

（3）Broadcast Receiver：Broadcast Receiver 不提供与用户进行交互的表示层，是一种不执行任何任务，仅仅是负责接收广播消息并对消息做出响应的组件。广播是传递消息的机制，对外部事件进行过滤。广播接收器没有用户界面，通过启动 Service 或 Activity 来响应它们接收到的信息，或者用 Notification Manager 来通知用户。

（4）Content Provider：应用程序通过 Content Provider 访问其他应用程序的一些私有数据，这是 Android 提供的一种标准的共享数据的机制。它屏蔽了内部数据的存储细节，向外提供了统一的接口模型，大大简化了上层应用的书写，对数据的整合提供了方便的途径。

2 网络更新开发的关键技术

Android 网络更新功能的开发过程中，需要用到很多关键的技术。例如：XML 文件解析技术、SQLite 数据库存储。采用 TOMCAT 作为服务器，下面将对这些技术进行介绍。

2.1 XML 解析技术

XML 是一种可扩展标记语言，用于标记电子文件使其具有结构性，由于它本身的平台无关性、系统无关性和语言无关性，为数据的集成和交互提供了极大的便利。因此得到了广泛的应用和支持。所谓 XML 文件的解析是指：把代表 XML 文档的一个无结构的字符序列转换为满足 XML 语法的结构化组件的过程[4]。Android 平台提供了 3 种解析 XML 文件的技术，分别为：DOM、SAX 和 PULL。本文中采用 PULL 解析器进行解析。

PULL 解析器是一个开源的 Java 项目，Android 系统中集成了 PULL 解析器，无需添加任何的 jar 文件。系统本身使用的 XML 文件，在内部解析时都是采用 PULL 解析器解析的。PULL 解析器的运行方式与 SAX 解析器相似，也是基于事件驱动的。在解析过程中，通过 parser.next()可以进入下一个元素并触发相应的事件。

跟 SAX 解析不同的是：PULL 解析器读取 XML 文件后触发相应的事件调用方法返回的是一个数字，而不是方法。PULL 解析可以在程序中控制解析的位置。

2.2 SQLite 数据库存储技术

Google 为 Android 较大的数据处理提供了 SQLite 数据库，SQLite 是一款非常流行的轻量级的嵌入式数据库，支持统一的 SQL 语句，并且突出的优点是占用非常少的内存资源，大约只占用几百 KB 的内存。Android 在运行时集成 SQLite，每个应用程序都可以使用 SQLite 数据库。SQLite 具备如下的特征：

（1）轻量级：SQLite 只需要带一个尺寸很小的动态库，就可以享受它的全部功能。与传统的数据库引擎采用 C\S 模式不同，它是进程内的数据库引擎，因此不存在数据库的客户端和服务器。

（2）独立性：SQLite 数据库的核心引擎本身不依赖第三方软件，使用时不需要安装，还支持多线程访问。

（3）多语言接口：SQLite 支持许多语言编程接口，比如 C\C++、Java、Python、Ruby、Perl 等，以及支持 ODBC 接口。

（4）隔离性：数据库中的所有信息，包括表、视图、触发器等都包含在一个文件内，方便管理和维护。

（5）安全性：SQLite 通过数据库的共享锁和独占性来实现独立事务的处理。在同一或线程在执行 SQLite 数据库的写操作之前，必须获得独占锁且其他的读或写操作将不再发生。

2.3 多线程技术

当应用程序启动后，Android 会启动一个相对应的主线程（又叫 UI 线程），用来负责处理与 UI 相关的事件，包括处理用户的按键响应事件、绘制屏幕事件等等。在开发过程中需要保证主线程时刻能够响应用户的操作，因此，有些操作需要另开一个线程来实现。例如：游戏中大量资源的加载、不同角色的控制等等。多线程技术是应用开发的关键技术。

在开发复杂的应用程序时都会使用到多线程技术。多线程技术是指在一个应用程序中，同时运行多个不同的线程，每个线程执行不同的任务。因此，使用多线程有如下优点：

占用系统资源少。线程又称为轻量级的进程，多个线程之间共享同一块内存空间和相同的系统资源，每个线程占用的资源非常小，便于开发。

提高用户体验。使用多线程进行开发，可以把不同的事件交给不同的线程进行处理，减少与用户的交互时间，提升用户体验。

避免程序出现无法响应的状况。因为 UI 线程的响应时间很短，当超过 5s 左右程序无法得到响应，UI 线程就会被阻塞，从而弹出应用程序无响应的对话框。当需要进行一些耗时的操作，例如：连接服务器，下载音乐等需要长时间等待的操作。采用多线程技术，可以很好的避免 UI 线程的阻塞，保证程序的良好运行。

在 Java 中实现多线程的方法有两种，第一种是实现 Runnable 接口创建线程，实现接口中的 run()方法，在 run()方法中处理具体的逻辑。第二种方法是继承 Thread 类创建线程，重写

Thread 类中的 run()方法，然后调用 start()方法启动这个线程。

2.4 TOMCAT 服务器

TOMCAT 是 Apache 软件基金会（Apache Software Foundation）的 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其他一些公司及个人共同开发而成[5]。目前版本支持最新的 Servlet 和 JSP 规范。

TOMCAT 是一个轻量级的应用服务器，且是一个 JSP/SERVLET 容器。它运行时占用的系统资源很小，具有良好的扩展性，支持负载平衡与邮件服务等开发应用系统常用的功能，而且还在不断的改进和完善中，开发人员可以更改或在其中加入新的功能。

TOMCAT 具有复杂而又模块化的结构，图 1 所示为 TOMCAT 的容器模型，由图中可知 TOMCAT 的核心是两大组件：Connector 和 Container。Connector 的主要任务是负责接收客户端发送过来的连接请求，创建一个 Request 和 Response 对象分别用于和请求端进行数据交换，然后产生一个线程来处理这个请求，并把产生的 Request 和 Response 对象传给这个线程。一个 Container 可以选择对应多个 Connector,多个 Connector 和一个 Container 可以形成一个 Service，从而对外提供服务。整个 TOMCAT 的生命周期由 Server 控制。

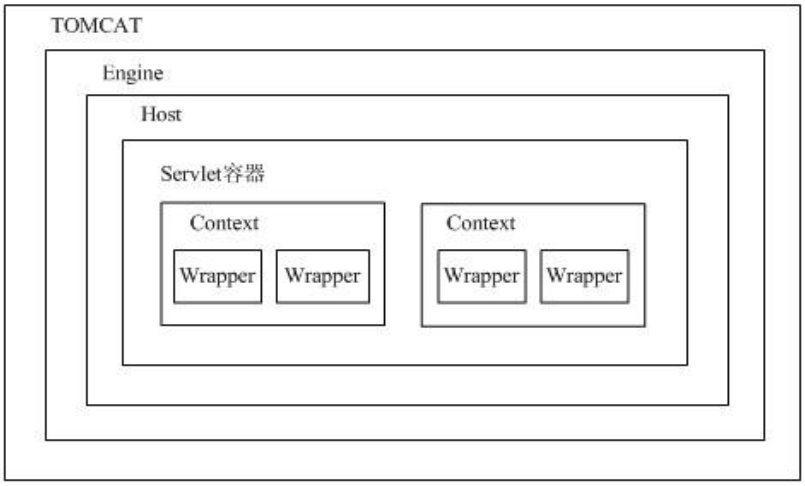

图 1 TOMCAT 容器模型图

Fig.1 TOMCAT container model

TOMCAT 作为 Servlet 容器，有 3 种工作模式：进程内的 Servlet 容器、进程外的 Servlet 容器和独立的 Servlet 容器。TOMCAT 默认的工作方式是独立的 Servlet 容器，它是内置在 WEB 服务器中的一部分，指的是使用基于 Java 的 WEB 服务器。

3 网络更新功能的设计与实现

3.1 客户端与服务器的通信原理

根据本应用的需求分析，需要为应用添加网络功能。服务器端选用 TOMCAT 作为远程服务器，因此，采用 C/S 客户端-服务器模式进行网络功能设计。使用 HTTP 协议作为互联网的的通信协议。

HTTP 协议是一种属于应用层的通信协议，它是面向事务的，即一系列的信息交换。HTTP 协议具有如下特点：

（1）支持 C/S 客户端-服务器模式。

（2）具有操作简单、响应快速、灵活的特点。当客户端向服务器发出请求时，只需要向服务器端发送请求的路径和请求的方法即可。每一种请求的方法都规定了客户端与服务器联系的类型，请求方法包括：GET、POST、HEAD 等。正是因为协议简单，因此使用 HTTP 的程序很小，响应的速度会很快。灵活是指传输的数据对象可以是任意类型。

（3）无连接。每一次客户端向服务器端发出连接请求时，只处理一个请求，服务器完成请求后，当得到客户端的应答后，连接就断开。这样保证了只需要很短的传输时间。

（4）是一种无状态的协议。即 HTTP 协议对于事务处理没有记忆功能[6]。缺少状态使得需要前面信息处理后续问题时，必须重传，重传将导致连接的次数和传输的数据量增大。

客户端使用统一资源定位符 URL 对服务器端发出请求，HTTP 的 URL 的一般形式为：http://<主机>:<端口>/<路径>。HTTP 协议的默认端口号是 80,URL 中的字母不分大小写。

HTTP 协议是支持客户端-服务器模式的，连接过程是：首先，客户端向服务器发出连接请求，当连接建立后，发送一个服务请求给服务器，请求方式为：统一资源定位符 URL、协议版本号等，当服务器接到请求后，将给客户端一个相应的响应信息，格式为：一个状态行，包括信息的协议版本号，一个成功或错误的代码等等内容。客户端与服务器端的交互主要分为 4 个过程：客户端与服务器建立连接、客户端向服务器发送服务请求信息、服务器端发送响应信息、关闭连接。如图 2 所示。

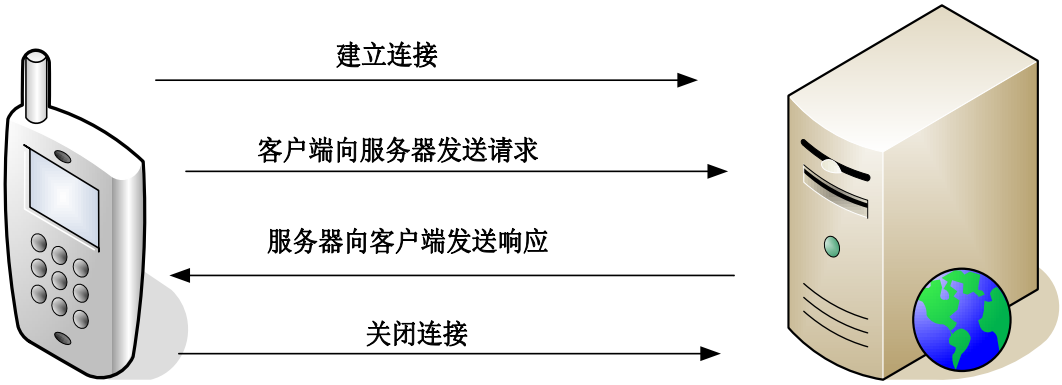

图 2 基于 HTTP 协议的通信过程

Fig.2 The communication process based-on HTTP protocol

3.2 网络更新功能的总体设计

网络更新功能的总体设计思路是：当应用发布或升级后，在服务器上放置最新的应用 APK 以及最新的版本号。版本号的存储采用 XML 文件的形式。客户端获取手机上已安装的应用版本号，并向服务器发出检测请求，连接网络解析出服务器上 XML 文件中的版本号，对两个版本号进行对比，如果服务器端的版本号大于当前的版本号，则提醒用户进行下载，下载时再次向服务器端发出下载请求，从服务器端下载最新的 APK，并完成安装。

在开发过程中，在服务器端放置应用的 APK 文件和版本号存储文件，考虑到管理和维护问题，每一个应用对应的 APK 和版本号放在同一个文件夹下。然后在客户端上，主菜单设置界面添加网络更新按钮，点击更新按钮实现网络功能。总体设计如图 3 所示。图中服务器上存放了一个 Thunder 文件夹，文件夹下存储了 Thunder.xml 文件和 Thunder.apk 文件。

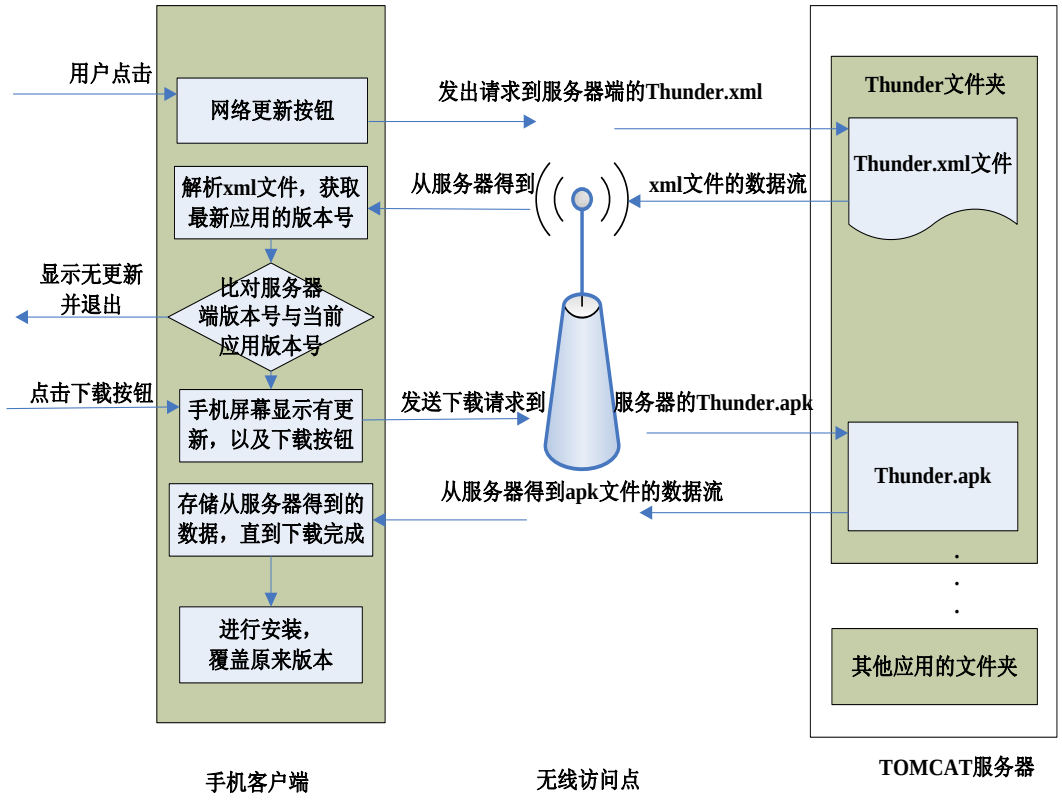

图 3 网络更新功能总体设计图

Fig.3 The whole design of updating function through network

3.3 服务器端版本的设计与实现

根据 XML 的特点，服务器端采用 XML 文件的形式存储版本号。

XML 是一种可扩展标记语言，可以用来创建属于自己的标记，信息描述时更侧重于结构化的描述。XML 是互联网应用中通用的一种数据通信格式，结构清晰，便于阅读和维护。

在更新版本时,在服务器上放置应用的 XML 文件，命名的方式为：应用名.xml。因此，这里取名为 Thunder.xml。XML 文件的内容为：

```
<?xml version="1.0" encoding="UTF-8"?>
    <apk>
        <version>1.1</version>
        <apkurl>http://192.168.1.110:8080/apk/Thunder.apk</apkurl>
        <description>有最新版本，请及时更新！</description>
    </apk>
```

XML 文件中设定 encoding 为 UTF-8 编码，是为了下面的描述可以使用中文。自定义内容时，首先定义一个标签，起始标签为<apk>，终止标签为</apk>。然后在大标签下定义 3 个子标签，<version>表示当前 APK 的最新版本号，如定义中的 1.1；<apkurl>子标签表示应用最新的 APK 的下载地址，当客户端解析出此标签，可以得到 APK 的下载链接进行下载；<description>标签表示的是对本应用的描述，为客户端提供一些提醒描述。需要注意的是：起始标签和终止标签是成对出现的。

3.4 版本检测功能的实现

应用网络更新的第一部分是检测是否有最新的版本需要更新。分别获取当前应用的版本号和服务器上 xml 文件中的版本号，根据比对版本号进行相应的操作。具体的操作流程如图

4 所示。

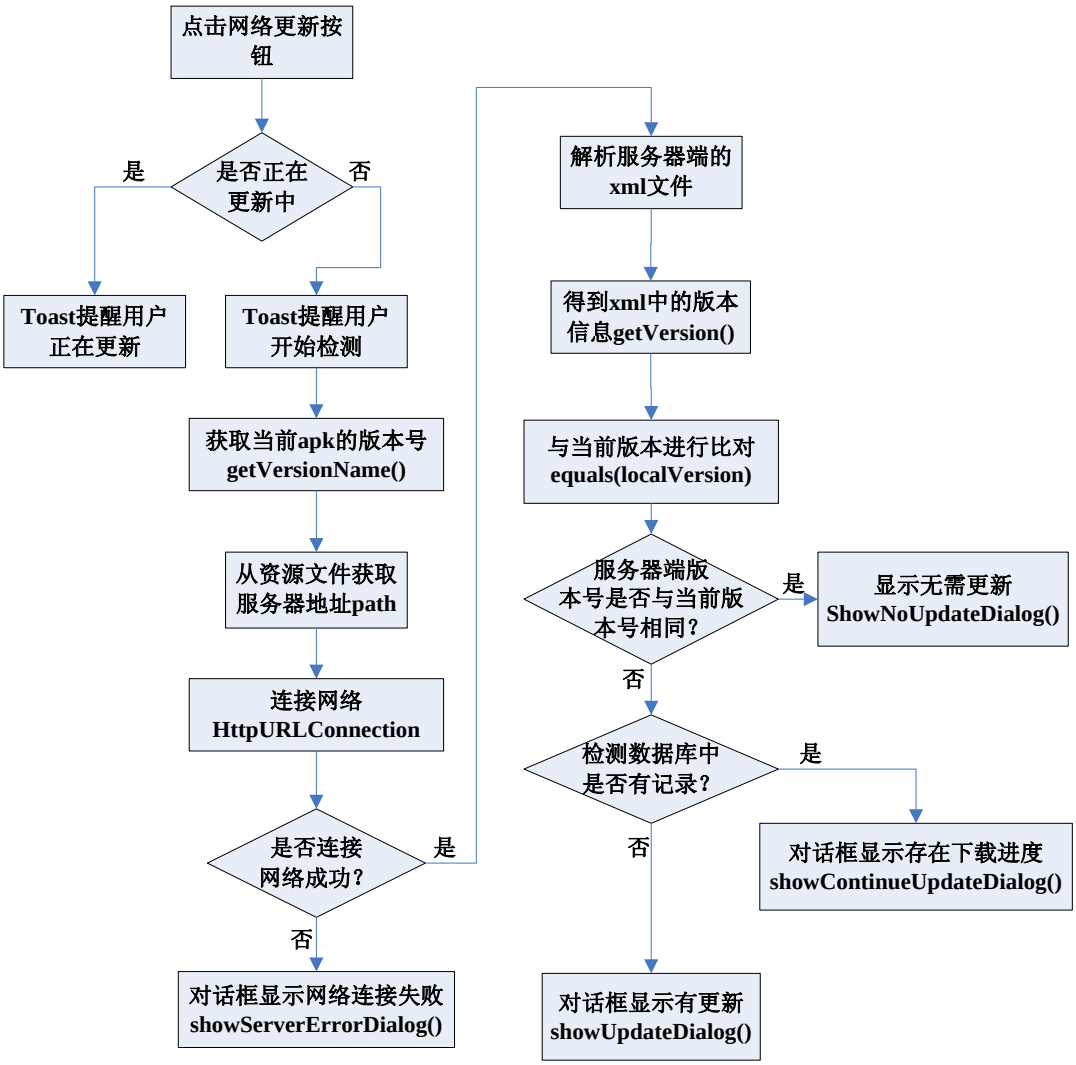

图 4 版本检测的操作流程

Fig.4 The operational process of version detection

版本的设置是在配置文件 AndroidManifest.xml 中，修改属性 android:versionName="*1.0*"。初始版本为 1.0。

版本检测在按下网络更新按钮开始，下面是具体的实现过程：

（1）先判断是否处于正在更新状态。

设置变量 isDownload，初始化为 false，当正在下载更新时，把 isDownload 置为 true。如果处于正在检测更新状态时，使用 Toast 方式提醒用户。

Toast.*makeText*(myActivity,R.string.*download*,Toast.*LENGTH_LONG*).show();

反之，isDownload=false，则开启线程检测版本号。

（2）检测当前应用的版本号。

定义 getVersionName()函数，返回 apk 的版本号。利用 Android 的 API-PackageManager 类，查询已安装应用的信息，调用 packagManager 的 getPackageInfo()方法获得 versionName。此应用的初始版本为 1.0。

（3）获取服务器端 xml 版本号。

从资源文件获取服务器的地址 http://192.168.1.110:8080/apk/Thunder.xml，把 path 包装成 url 对象，使用 HttpURLConnection 创建连接对象 conn，设定连接的超时时间为 4000 毫

秒。获取连接对象 conn 的输入流，即 conn.getInputStream()，得到 xml 的数据内容。再通过 xml 文件解析技术，从解析的 xml 中得到更新信息对象 updateInfo，从而调用 getVerson()获取最新的版本号。

解析服务器端 Thunder.xml 文件采用 PULL 解析器进行解析。xml 文件结构简单，而 PULL 解析器速度快，占用内存小，并且代码简洁。创建解析类 UpdateInfoParser，定义静态函数 getUpdateInfo(InputStream is)，返回值为 UpdateInfo，即为获取消息对象提供接口。首先创建 XmlPullParser 的对象：XmlPullParser paser=Xml.newPullParser()；把获得的输入流 inputStream 当作参数传入，设置解析的数据源。使用 getEventType()获取解析时间的各个类型：START_DOCUMENT、END_DOCUMENT、START_TAG、END_TAG、TEXT。用一个 while 循环判断是否到达解析文件的 END_DOCUMENT 事件，未到达文件尾部时，判断 xml 文件中的起始标签，是否与 version、apkurl、description 相同，相同时，获取此标签的内容直到终止标签。

（4）比对版本号大小。

使用 equals 判断 localVersion 与网络服务器上存的版本号的大小关系，如果二者相同，调用 showNoUpdateDialog()函数，弹出对话框提醒用户，已经是最新版本，无需升级。如果二者大小不同，则需要判断是否已经存在下载进度，即接下来是进行断点续传还是第一次开始下载。通过步骤（3）中获得的更新信息对象 updateInfo 获取要下载 APK 的 url 地址，即 updateInfo.getApkUrl()，并对话框提醒用户进行升级版本。

3.5 下载功能的实现

当检测到了有最新的版本时，用户点击更新对话框的确定按钮，开始从服务器端下载 APK。为了下载的同时不影响用户对当前应用的操作，使应用有更好的用户体验，把下载功能放在后台下载，即使用组件 Service 实现后台线程池下载。同时把下载的进度在消息通知栏显示，实时更新下载进度。

组件 Service 是 Android 开发中重要的组件。Android 应用开发中必须遵循单线程模型的原则，UI 操作不是线程安全的。如果把耗时操作放在 UI 主线程执行，UI 线程将无法及时的分发事件，也不能及时的更新绘制界面，将导致 UI 线程被阻塞。并且在设计 Android 时，规定如果应用在 5s 内无法得到响应的话，系统将弹出应用无响应 Application is Not Response 对话框。这样的响应是需要开发者避免的。因此，在本应用设计时考虑这个因素，把网络的连接、下载放在一个子线程处理。

下载时采用线程池技术进行下载，提高下载速度。Java 的 JDK1.5 为开发者开发并发程序提供了工具包 java.util.concurrent，利用工具包提供的创建线程池的方法，能有效的避免线程死锁问题和减少竞争条件。考虑到本应用下载时间以及手机的内存资源等因素，使用 newFixedThreadPool()方法进行下载，并设置线程数为 5。当线程池执行完成时，可以调用 ExecutorService 类中的 shutdownNow()方法，将强行关闭 ExecutorService 管理，取消一切运行中和在共享队列中等待的任务。

使用组件 Service 进行后台下载时，首先需要在应用配置文件中添加网络访问权限和存储访问权限等。添加如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
```

android:name="*android.permission.MOUNT_UNMOUNT_FIFESYSTEMSE*"/>
<uses-permission android:name="*android.permission.MODIFY_AUDIO_SETTINGS*" />
此外，在配置文件中注册服务，即工程中建立的 Service 类，如下：
<service android:name="*com.cj.httpservice.ApkService*"></service>

应用下载之前需要调用系统函数 Environment.getExternalStorageState()获取存储状态，判断是否为 MEDIA_MOUNTED。当用户点击下载按钮时，创建 Intent 对象，把 xml 文件中 apk 的下载地址打包成键值对的形式，intent.putExtra("apkurl", updateInfo.getApkUrl())；采用 startService(intent)的方式启动 Service 服务。使用这种方式启动 Service，无论调用 startService() 的次数多少都不会导致重复创建服务，服务与其调用没有关联，即使程序退出了，调用的服务仍然处于运行状态。

Service 创建后调用其 onCreate()方法，在 onCreate()方法中创建 Handler 对象，这个 Handler 是用于给通知栏发送消息，实时的发送下载的状态。然后调用 Service 的 onStartCommand 方法，在此方法中为下载做准备工作。一方面设置通知栏的属性，添加下载任务到通知栏。使用 getSystemService()系统方法获得 NotificationManager 通知管理器的对象以及创建 Notification 对象，设定通知栏的下载图标和显示文本，调用 notification 的 setLatestEventInfo()函数显示通知栏的所有属性信息，最后将下载任务添加到任务栏中。另一方面为下载做初始化准备。使用 Executors.newFixedThreadPool(5)创建线程池 executorService 对象，如果存在下载进度，则从手机 SQLite 数据库中获取已经下载好的长度 completeLength。由于 apk 应用在服务器端，需要知道整个 apk 的大小，所以要连接网络获取。通过创建 URL 对象 url，使用 url 的 openConnection()方法建立 HTTP 连接对象，并建立网络连接，获取 apk 的大小 totalSize。

下载的准备工作完成后，通过实现 Runnable 接口创建一个新线程，把新线程加入到线程池调度。应用的保存路径为默认的存储路径，通过 Environment.getExternalStorageDirectory() 获得文件路径。建立网络连接时仍采用 HTTP 建立，获得 HTTPURLConnection 对象 httpConn，设定 http 的连接参数，包括连接超时时间、读取超时时间、请求的方式 GET、下载的起始和终止位置等。网络连接好后，得到 httpConn 的输入流，开始把得到的字节流写入文件中。使用文件随机存取类 RandomAccessFile 进行文件写入，调用 RandomAccessFile 的 seek()方法，找到文件中长度为 completeLength 的位置，从这个位置开始写入。使用 while()循环写入线程池中得到的字节流数据，直到下载完成为止。边写入的同时，需要判断写入的文件大小占总文件大小的百分比，如果比例是 5 的倍数，则使用 Handler 机制，sendMessage 到通知栏，通知通知栏进行进度更新。每次从缓冲区读取数据后，要及时的更新数据库中的文件记录，把当前的下载长度保存到数据库中。如果下载完成，则发出广播消息，进入安装过程。

在使用 Service 下载的同时，对消息的处理采用广播的方式通知。广播接收器的注册方式分为两种：代码注册和在配置文件 AndroidManifest.xml 中注册，在配置文件中注册的广播接收器是系统级的，将一直伴随着程序的存在。因此，在 AndroidManifest.xml 文件中注册广播接收器：install 和 installfail。广播的发送方式是：在需要发送广播时，把过滤信息和要发送的信息打包装入一个 Intent 对象，然后通过 sendBroadcast(intent)方法，把对象以广播的形式发送出去。当 intent 发送成功后，系统会判断此广播是否为标签<intent-filter>中注册的广播行为，如果是，则会调用广播的 onReceiver()方法进行处理。当广播执行完毕，广播对象也将被销毁。

APK 安装时，底层是通过系统服务 PackageInstaller 来识别安装的 Intent 对象，而

PackageInstaller 服务会在内部匹配 intent，MIME 类型 application/vnd.android.package-archive。具体的实现代码如下：

```
Intent intent = new Intent();
intent.setAction(Intent.ACTION_VIEW);
intent.setDataAndType(Uri.fromFile(file),"application/vnd.android.package-archive");
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
context.startActivity(intent);
```

3.6 断点续传功能的设计与实现

为了避免下载中断导致重新下载，设计了断点续传功能。其原理是保存好上一次下载的进度，下一次下载时找到前次下载的位置继续下载。因此需要保存前次的下载长度，根据 android 内置的 SQLite 数据库的特点，采用 SQLite 数据库的方式保存，创建的 SQLite 数据库是该应用独享的[7]。由于要保存的只是应用下载的路径和下载的长度，所以使用 SQLite 数据库占用的资源非常小，大概 5k 左右。设计时创建一个数据库，在数据库中建立存储的表，设计一个管理数据库的类。同时，设计一个数据库工具，对数据库进行操作，更新、删除、查询等。整个的设计类图如图 5 所示。

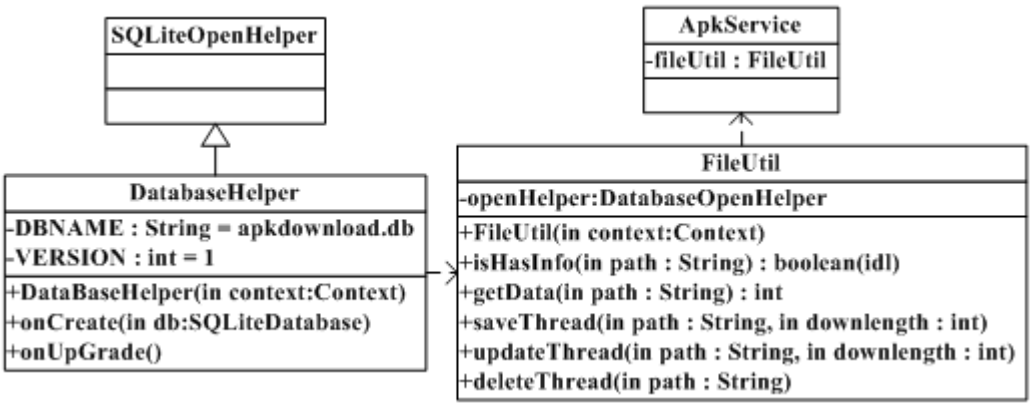

图 5 断点续传功能的类图设计

Fig.5 The design of class diagram for resume broken transfer

创建数据库管理类 DataBaseOpenHelper，继承自 SQLiteOpenHelper 类。在类中定义一个静态字符串常量 DBNAME=”apkdownload.db”，表示本应用建立的数据库名称。DatabaseHelper 类的构造函数中创建此数据库。在 onCreate()方法中创建数据库中的表，实现代码为：

```
db.execSQL("CREATE TABLE IF NOT EXISTS downloadinfo" +"(id integer primary key autoincrement, downpath varchar(100),downlength INTEGER)");
```

表 downloadinfo 中主码为自动增加的数据，两列为应用下载的路径、当前下载的长度。而 FileUtil 类为数据库的工具类，实现数据库表的查询、更新和删除操作。实现的函数为：

isHasInfo(String path):此函数为查看数据库中是否有此下载路径的下载记录。在判断是否存在下载进度时首先使用 getReadableDatabase()方法打开这个只读数据库。查询的 SQL 语句为：

sql = "select count(*) from downloadinfo where downpath = ?";

根据路径、sql 语句和 rawQuery()方法，返回 Cursor 对象，然后将 cursor 的光标通过 moveToFirst()方法移动到第一个位置，如果 Cursor 为空，则返回 false，表示无记录；反之，返回 true，表示有记录。

getData(String path):获取已经下载的文件长度，返回值为文件长度。方法同函数 isHasInfo()类似，找到游标移动到的第一个位置，使用 cursor.getInt(0)获取第一列的数据，此数据即为下载的长度。

saveThread(String path,int downlength):此函数为插入操作，即第一次下载时保存线程池已经下载的文件长度。使用数据库管理类对象的 getWritableDatabase()方法，使数据库处于可读写的状态，创建 SQLiteDatabase 的对象 db，通过 sql 语句把数据写入数据库表中。最后调用 db.setTransactionSuccessful()表示写入成功。

updateThread(String path,int downlength):此函数作用为实时更新已经下载的文件长度。实现过程与函数 saveThread()类似，只是 SQL 语句为：

String sql ="update downloadinfo set downlength=? where downpath=?";

经过上面的类和函数设计，最终实现了下载的断点续传功能。

4 结论

本文研究的是基于 Android 平台的网络更新功能的设计与实现。首先介绍了 Android 平台的相关概念，以及开发需要用到的关键技术，包括 XML 解析技术、SQLite 数据库存储、多线程技术等。然后详细的阐述了网络更新功能的设计与实现，其中先描述了网络更新的整体设计，然后分为版本检测、下载功能、断点续传功能 3 部分进行详细的设计与实现。通过本文的介绍，旨在帮助大家对 Android 应用的网络更新有更加深入的了解，开发人员可以通过本文的网络更新开发框架，针对自己的应用开发网络更新功能，从而更好的为用户提供服务。

[参考文献] (References)

[1] 李晓珊. 苹果 IOS、谷歌 Android、微软 Windows phone 三大移动互联网系统开发策略比较研究[J]. 中国广播，2013，10（5）：40-45.
[2] 邸烁. Android 揭秘和未来发展趋势[OL]. [2008-3-17]. http://publish.itpub.net/zt/android/index.html
[3] Ableson F. Using XML and JSON with android[OL]. [2010-08-24]. http://www.ibm.com/developerworks/library/x-andbenel/index.htm1
[4] 陈石. XML 技术及其应用[J]. 计算机应用研究，2002，19（3）：115-117.
[5] 张洪伟. Tomcat Web 开发及整合应用[M]. 北京：清华大学出版社，2006.
[6] 詹建飞. Android 开发关键技术与精彩案例[M]. 北京：电子工业出版社，2012.
[7] 邹丽丽. Android 若干关键技术研究与应用系统开发[D]. 杭州：浙江大学，2013.