Android多线程断点续传下载文件类设计

对于Android平台，很多网友可能考虑开发一个软件商店，对于Android平台上如何实现断点续传操作呢？这里Android123给大家一些思路和原理的介绍，同时在Android手机上要考虑的一些事情。

1. 流量控制，获取运营商的接入方式，比如说使用移动网络接入，尽可能的提示用户切换WIFI或提示，限制下载的流量以节省话费。

2. 屏幕锁控制，屏幕锁屏后导致应用会被挂起，当然Android提供了PowerManager.WakeLock来控制。

3. 对于断点续传，这要追溯到Http 1.1的特性了，主要是获取文件大小，如果这个无法读取的话，那么就无法断点续传了只能使用chunked模式了，当然获取远程服务器上文件的大小可以通过Http的响应头查找Content-Length。

4. 获取上次文件的更改时间，对于断点续传来说比较有风险的就是 继续下载的文件和早期下载的在server上有变动，这将会导致续传时下载的文件版本和原始的不同，一般有两种解决方法，早期我们配置服务器时通过Last-Modified这个http header获取文件上次修改时间，不过本次Android开发网推荐使用更为强大的ETag，ETag一般用于解决同一个URL处理不同返回相应，比如Session认证，多国语言，以及部分黑帽的SEO中。具体的实现大家可以参考RFC文档。

5. 考虑服务器的3xx的返回，对于专业的下载文件服务器会考虑到负载平衡问题，这就涉及到重定向问题，处理重定向使用Android的Apache库处理比较好。

6. 至于多线程，这里CWJ提示大家可能存在独立的线程下载一个文件，和多个线程分块下载单个文件之分，其中后者需要考虑上次下载数据是否存在问题，同时如果服务器不支持文件大小获取，则无法通过分段下载数据，因为不知道如何分段，所以在chunked模式中，只能使用一个线程下载一个文件，而不是多个线程下载一个文件。

7. 下载后的数据效验，可以考虑CRC等方式，当然对于一般的传输只要逻辑不出现问题，基本上不会有偏差。

8. 考虑DRM问题，这个问题在国内用的比较少，而国外的受数字保护的音乐和视频，需要额外的获取证书等。

9. 重试次数，对于一个文件可能在本次网络传输中受到问题，尤其是移动网络，所以可以设置一定的重试次数，让任务单独的走下去。

10. 线程开发方式，这里如果你的Java基础比较好，推荐直接使用Java并发库API比较好，如果过去只做过Java开发使用Thread即可，如果Java技术不过关可以Android封装的AsyncTask。