文章编号：2095-4980(2013)05-0718-06

Android 多媒体框架下 Stagefright 的功能扩展

温　伟，刘荣科

(北京航空航天大学 电子信息工程学院，北京 100191)

摘　要：鉴于现有 Android 系统多媒体框架的局限性，对其进行扩展，使之支持所有常见的多媒体文件格式和编码标准，以满足智能电视等应用需求。采用桥接设计模式，基于 FFmpeg 创建并添加了适配于 Android Stagefright 框架的解包装和解码插件，新插件支持的文件格式和编码标准齐全且支持本地文件、http 网络多媒体和实时流传输协议(RTSP)多媒体实时流 3 个信道。相对于传统扩展方案，基于插件的扩展方法使得新多媒体框架移植性好，Java 应用程序与原 Android 完全兼容。

关键词：智能电视；Android 系统；Stagefright 框架；FFmpeg 框架；多媒体框架

中图分类号：TN949.198；TP316.5　　**文献标识码：**A　　**doi：**10.11805/TKYDA201305.0718

Function extension for Stagefright of Android media framework

WEN Wei，LIU Rong-ke

(School of Electronic and Information Engineering，Beihang University，Beijing 100191，China)

Abstract：Due to the capacity limitation of Android media framework，it is extended to support all normal containers and codecs of multimedia, aiming to satisfy the demands of Smart TV. Based on bridge pattern and FFmpeg libraries，a container extractor component and a decoder component are created and plugged in，respectively. Every plug-in adapts to the architectures of both Stagefright and Android media framework. The experimental results show that the extended Android can support all normal containers and audio/video codecs; moreover, it can parse multimedia streaming from local file system, http and Real Time Streaming Protocol(RTSP). Compared to traditional methods, this plug-in based design pattern promises the extended Android with portability and the compatibility with its Java applications.

Key words：smart TV；Android；Stagefright；FFmpeg；media framework

随着手机处理能力和移动通信的蓬勃发展，嵌入式智能操作系统也取得了飞速发展，其中以苹果 iOS 系统、谷歌 Android 系统和微软 Windows Phone 7 系统为代表。Android 系统由于其开源性和开放性，在近 5 年内取得了最快的增长，目前占用全球智能手机的 59%份额，并且每天新增 900 000 个 Android 设备。与此同时，一方面，随着移动带宽和终端处理能力的发展，手机用户对多媒体类型的需求日益增长；另一方面，借鉴于智能手机取得的成功，将 Android 系统移植到数字电视机顶盒上[1]，构建类似于智能手机的生态环境，为数字电视广播网和电视终端赢得用户，已成为数字电视广播行业的研究热点，“智能电视”的概念也应运而生，智能电视的主要业务之一为多媒体业务[2]，其要求 Android 系统多媒体框架支持所有常见的多媒体文件格式和编码标准。这两方面均对现有 Android 系统提出了挑战，因为 Android 系统最初为普通手机定制，其多媒体框架能力有限。因此，扩展 Android 多媒体框架具有很大的实用价值。

1　Android 多媒体框架

Android 软件栈从底至顶分为 4 层[3]：1) Linux 内核，包括 Google 添加的特有驱动；2) 本地库和 Android 运行库，包括标准 libc、多媒体框架本地库、数据库 SQLite、浏览器 WebKit 等，Android 运行库是个 Dalvik 虚拟机，完成 java 层到本地库的调用等；3) java 框架，将下层服务和功能进行封装，供应用程序开发者调用；4) 核心应用，基于 java 框架，Android 开发了一些自带的核心应用，包括启动器、电话模块、通信录和浏览器等。

收稿日期：2012-08-14；修回日期：2012-10-22

多媒体框架位于第 2 层，是一个本地库。该框架下的 Stagefright 将音视频的包装格式解析、编解码和显示等进行封装，提供对整个多媒体信息处理的支持。音频、视频及其他数据借助于某个特定的形式被组织和复用到同一文件或多媒体流中，形成某个特定的包装格式，又称为容器(Container)。最初 Android 多媒体框架采用了 OpenCore，由于其复杂性，自 Android 2.3，完全采用了 Stagefright，其中 Android 2.3.3 是目前市场应用最为广泛的 Android 版本，65%的 Android 设备采用了 Android 2.3.3，本文研究内容也基于 Android 2.3.3。

1.1 Android 多媒体框架的软件架构

Android 多媒体框架见图 1，它贯穿了 Android 软件栈的 4 个层次。

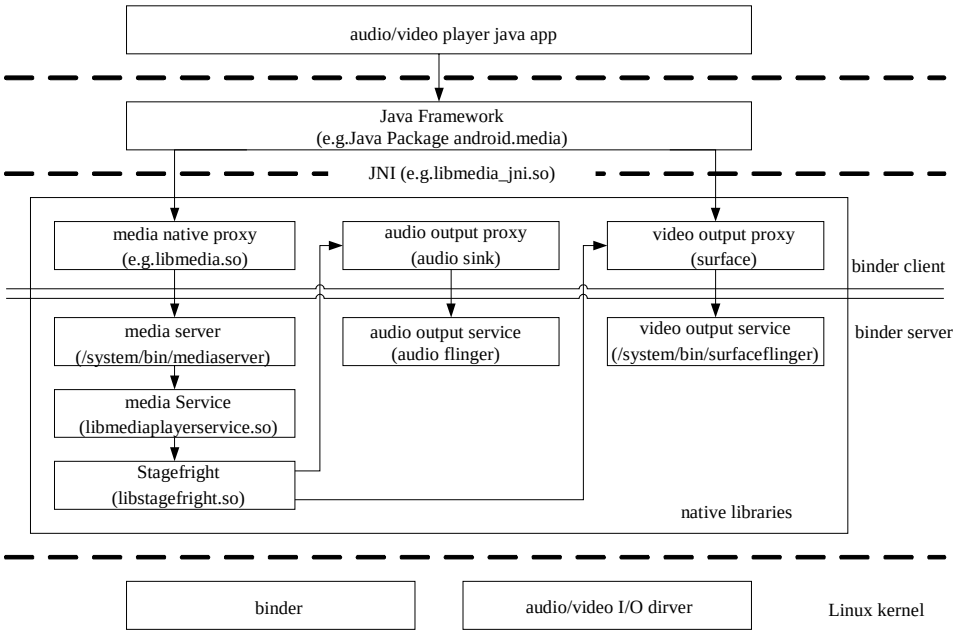


Fig.1 Media framework of Android
图 1 Android 多媒体框架

如图 1 所示,以音视频的播放为例,顶层的应用程序基于 Java Framework 开发,向下通过本地接口(Java Native Interface，JNI)调用多媒体服务的本地客户端代理(Media Native Proxy，MNP)完成播放，该代理通过 Binder IPC(Inter Process Communication，IPC)机制获取多媒体服务。

Binder 为 Android 系统 Linux 内核特有的驱动，提供进程间通信机制，Binder IPC 基于客户端/服务器(C/S)的架构，客户端通过 Binder 调用服务端接口就像访问本地接口一样，Binder 驱动隐藏了所有进程间数据交互的实现细节。Binder 机制非常类似于分布式计算里的公共对象请求代理结构(Common Object Request Breaker Architecture，CORBA)[4−5]。

如图 1 所示,Media Native Proxy 作为客户端通过 Binder 访问进程 mediaserver 提供的多媒体服务。mediaserver 启动后作为多媒体服务的宿主进程，会调用多媒体本地库(如，libmediaplayerservice.so 等)，进而调用 Stagefright。Stagefright 完成多媒体源的解包装和解码后，通过音视频输出代理(Audio Output Proxy 和 Video Output Proxy)访问音视频输出服务(Audio Output Service 和 Video Output Service)，完成播放显示。

1.2 Android 多媒体框架下的 Stagefright 模块

Android 多媒体框架下的 Stagefright 如图 2 所示，其中虚线箭头表示通过 Binder 间接访问，本文 Stagefright 扩展方法需修改的模块以粗体直角框表示，需添加的模块以粗体圆角框表示，其他模块不是本文的研究重点。

Stagefright 主要包括了 AwesomePlayer,MediaExtractor,OMXCodec,AwesomeRenderer 和 AudioPlayer 等[6]，其中 AwesomePlayer 作为其他对象的组合，可依次通过 DataSource 获取多媒体源的数据，再通过 MediaExtractor 解包装插件完成包装格式解析，并提取音视频流，进而通过 OMXCodec 调用解码插件完成音视频流的解码，最后通过 AwesomeRenderer 和 AudioPlayer 将原始数据送往 I/O 设备，完成播放显示。

MediaExtractor 作为解包装插件的插槽，会逐一访问各解包装插件的嗅探(sniff)接口，以选取当前多媒体源

的最佳解包装插件。Android 2.3.3 的解包装插件有 MPEG4Extractor,MP3Extractor,AMRExtractor,WAVExtractor, OggExtractor,MatroskaExtractor 和 MPEG2TSExtractor，每个解包装插件都有一个对应的嗅探接口。以 MPEG4Extractor 为例，其为 MPEG4 的解包装插件，嗅探接口为 SniFFmpeg4，SniFFmpeg4 会根据 MPEG4 文件特定字段评估该多媒体源为 MPEG4 包装格式的概率。MediaExtractor 每调用一个嗅探接口便会得到一个信心值，最终选取信心值最高的插件作为解包装插件。显然，Android 解包装插件非常有限，如常见的 avi,mkv,mov 和 wma 等包装格式均不支持。本文基于 FFmpeg 新增一个多功能解包装插件 FFmpeg Extractor 及其对应的嗅探接口 SniffFFmpeg，达到扩展 Stagefright 解包装的目的。

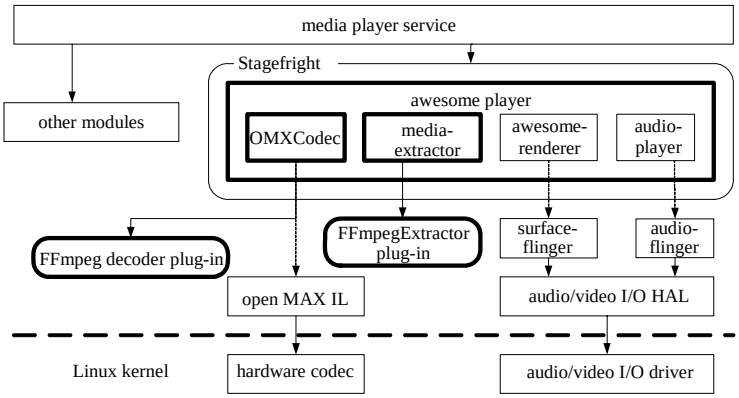


Fig.2 Stagefright module of Android

图 2 Android Stagefright 模块

解包装插件从多媒体源获知音视频流编码信息后，解码插槽 OMXCodec 将据此搜索可用的解码插件。Android 2.3.3 的解码插件包括 MP3Decoder,AMRNBDecoder,AMRWBDecoder,AACDecoder,G711Decoder,M4vH263Decoder,AVCDecoder 和 VPXDecoder。显然，Android 解码插件比较有限，如 MPEG2,WMV,WMA 和 AC3 等音视频编码标准均不支持。本文基于 FFmpeg 新增一个多功能音视频解码插件 FFmpegDecoder，达到扩展 Stagefright 解码的目的。

2 基于 FFmpeg 的 Stagefright 功能扩展

在众多多媒体包装格式和编解码标准中，急需一套完备的解包装和解码框架，FFmpeg 便是这样一套框架。本文借助了 FFmpeg 提供的 4 个库：1) libavutil，包括随机数发生器、数据结构、数学运算和核心多媒体实用工具等；2) libavcodec，音视频编解码标准的编码器和解码器；3) libavformat，多媒体包装格式的复用和解复用；4) libswscale，高度优化的图像缩放和色空间格式转换库。本文 FFmpeg 版本为 0.9.1。

基于 FFmpeg 扩展 Android 多媒体框架的传统方法有 2 种：

1) 基于 FFmpeg 和 Android NDK 进行扩展。本方法开发流程简单，无需关心 Android 系统多媒体框架，因此也是应用较广泛的一种方法。基于 NDK 的应用程序直接调用本地库，程序执行效率较高，但兼容性和移植性均很差，与 Android 开放的生态环境相悖，Google 不推荐基于 NDK(Native Development Kit)进行应用开发。

2) 基于 FFmpeg 完全实现一套新的多媒体框架。文献[7]便基于本方法，设计者无需关心 Android 系统原多媒体框架，但需完全实现一套新框架的本地库、JNI,Java Framework 和应用，方案复杂而冗余。此外，Java Framework 向上提供的接口发生了变化，基于新框架开发的应用程序无法与原 Android 系统兼容，此方案不可取。

本文不采用上述 2 种方法，而是基于 FFmpeg 新增适配于 Stagefright 的解包装和解码插件。本方法在深入理解 Stagefright 框架的基础上，只需新增 2 个插件并对 Stagefright 作少量配置。插件设计模块化，符合良好的软件设计模式，移植性好。特别地，此方法未对 Java Framework 做任何修改，基于原 Android 系统接口开发的应用能直接在新框架上运行，兼容性好。3 种方案的比较见表 1。

表 1 Android 多媒体框架功能扩展方案比较

Table1 Differences among function extension methods for Android media framework

methods	framework portability	app compatibility	complexity	efficiency
NDK based	bad	bad	low	high
create a new framework	bad	bad	high	applicative
Stagefright plug-in based	good	good	medium	applicative

Android 软件栈设计模式良好，实现新的解包装和解码插件只需实现特定抽象类的接口。新插件实现后，将

它们添加到备选插件列表中即可完成插件植入。图 3 为本文新插件 FFmpegExtractor 和 FFmpegDecoder 的 UML(Unified Modeling Language)类图，图中 sp 为 Android 强引用模板，防止忘记内存释放而导致内存泄露。后文将以解包装插件实现、解码插件实现和插件植入的顺序阐述具体细节。

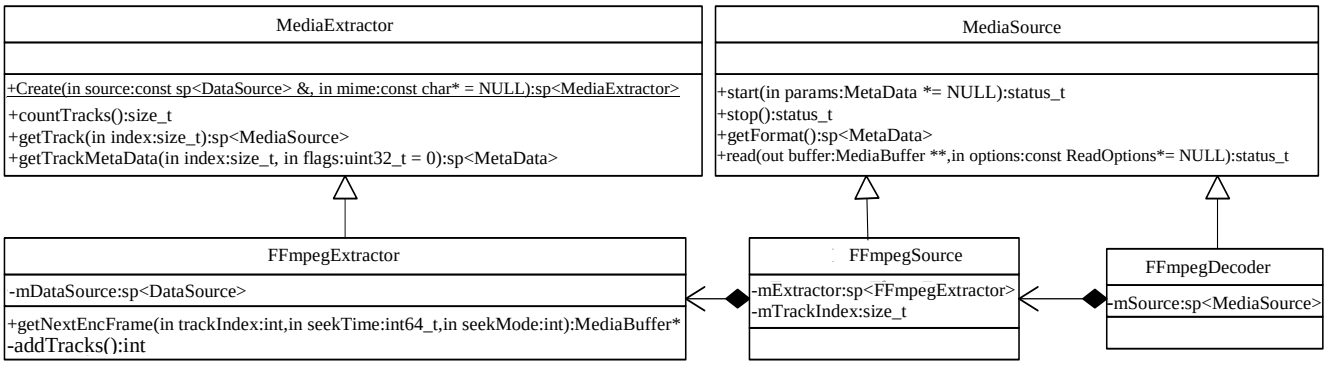


Fig.3 UML class diagram of FFmpeg extractor plug-in and FFmpeg decoder plug-in
图 3 FFmpegExtractor 和 FFmpegDecoder 的 UML 类图

2.1 实现适配于 Stagefright 的多功能解包装插件

无一例外地,Android 解包装插件均实现了 MediaExtractor 抽象类的接口,这些接口包括 countTracks、getTrack 和 getTrackMetaData 等，因此 FFmpegExtractor 插件也需实现这些接口，如图 3 所示。FFmpegExtractor 插件中的 mDataSource 保存了多媒体源的地址，地址可能为文件路径、http 网络文件地址和 rtsp 实时流地址等。

图 3 中几个 FFmpegExtractor 重要操作的实现方法为：

1) addTracks：本操作在 FFmpegExtractor 构造时执行，主要基于 FFmpeg 找到并保存 mDataSource 地址上多媒体源的音视频流相关信息和上下文环境等。上下文环境指 FFmpeg 库中解包装和解码的上下文环境，以 AVFormatContext 的形式组织，几乎所有 FFmpeg 操作均基于此，其中保存了大量音视频流相关信息，如视频分辨率、音频采样率和编码类型等，其中较为重要的信息是音视频的编码类型，OMXCodec 将据此选择合适的解码插件。

2) getTrack：申请一个 FFmpegSource 并返回。如图 4，由于解包装插件在 AwesomePlayer 中以 MediaSource 的形式存储，但是，如图 3 所示，FFmpegExtractor 是一个 MediaExtractor 而不是 MediaSource，因此设计了一个实现 MediaSource 的 FFmpegSource，并将 FFmpegExtractor 存储其中。在多媒体播放过程中 AwesomePlayer 将通过本接口获取一个解包装插件的适配器，FFmpegSource 充当了 FFmpegExtractor 的适配器。

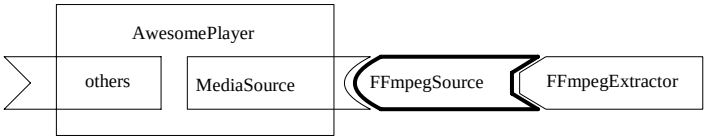


Fig.4 FFmpegSource adapter
图 4 FFmpegSource 适配器

3) getNextEncFrame：此接口为插件真正的解包装接口，将基于 FFmpeg 对多媒体源进行解包装并返回音视频帧数据。为了使得音视频播放时能够同步，需根据 FFmpeg 提取的显示时间戳(Presentation Time Stamp，PTS)信息等得到 Android 多媒体框架下的时间戳。最后，帧数据和时间戳一起返回。

FFmpegSource 适配器的设计很简单。在后文的解码插件设计中，解码插件将通过 FFmpegSource 的 read 接口获取音视频流，因此 read 只需调用 FFmpegExtractor 的 getNextEncFrame 来实现。

2.2 实现适配于 Stagefright 的多功能解码插件

Android 解码插件均实现了 MediaSource 抽象类的接口，因此 FFmpegDecoder 插件也需实现这些接口，如图 3 所示。FFmpegDecoder 插件中的 mSource 将保存解包装插件的适配器，如 FFmpegSource。由于 FFmpegDecoder 插件的实现原理与 FFmpegExtractor 类似，此处仅作简单阐述。FFmpegDecoder 构造过程需进行 FFmpeg 注册并初始化 mSource；start 完成 FFmpegDecoder 的初始化准备工作；read 为 FFmpegDecoder 的核心解码接口，通过 mSource 的 read 解包装得到音视频流后利用 FFmpeg 开始解码；stop 主要释放 start 中申请的资源。

如何基于 FFmpeg 解包装并解码，可参考文献[8]。

2.3 新插件的植入

实现新插件后，接下来的工作便是将其植入到 Android 系统中。

解包装新插件的植入主要配置 MediaExtractor 的静态接口 Create，此静态接口逐一访问各个解包装插件的嗅探接口，返回信心值最高的解包装插件，因此需将 FFmpegExtractor 的嗅探接口 SniffFFmpeg 添加到嗅探列表中，这只需在 DataSource 静态接口 RegisterDefaultSniffers 下添加“RegisterSniffer(SniffFFmpeg)”。可以灵活设置 SniffFFmpeg 返回的信心值，由于 FFmpeg 几乎支持所有常见包装格式，本文去掉了 Android 原来所有解包装，直接选用 FFmpegExtractor 进行解包装。

类似地，解码插件通过 OMXCodec 的静态接口 Create 进行搜索。Create 通过解包装适配器(如 FFmpegSource)获取解码器类型，根据解码器类型在 kDecoderInfo 中搜索匹配的解码插件名字，最后从 kFactoryInfo 中搜索匹配的解码插件。因此新解码插件的植入只需配置 3 处。

表 2 FFmpeg 解码器类型的定义

Table2 Definition of FFmpeg decoder MIME

decoder MIME	value	plug-in decoder name
MEDIA_MIMETYPE_VIDEO_FFMPEG	"video/FFmpeg"	"FFmpegDecoder"
MEDIA_MIMETYPE_AUDIO_FFMPEG	"audio/FFmpeg"	"FFmpegDecoder"

1) 如表 2 第 1 列和第 2 列，在 MediaDefs 中添加音视频的 FFmpeg 解码器类型；

2) 如表 2 第 1 列和第 3 列，将编码器类型与新插件名字的对应关系增加到 kDecoderInfo；

3) 通过“FACTORY_REF(FFmpegDecoder)”将新插件添加到 kFactoryInfo 列表中，并通过“FACTORY_CREATE(FFmpegDecoder)”定义插件新建方法。

Android 有一套自己的编译系统，编译时会依次查找各文件夹下的 Android.mk 文件，并根据 Android.mk 规范对各个模块进行编译。最后，将 FFmpeg 源码、FFmpegExtractor,FFmpegSource 和 FFmpegDecoder 添加到 Android 源码适当目录，并添加和修改一些 Android.mk，即可完成多媒体框架的扩展。本文先基于 Android 2.3.3 模拟器完成了多媒体框架扩展，将新多媒体框架移植到机顶盒等平台非常简洁，只需将 FFmpeg 动态库 libavutil.so, libavcodec.so,libavformat.so 和 libswscale.so 添加到/system/lib 目录并替换掉/system/lib/libstagefright.so。

3 实验结果

本文对 Stagefright 扩展后的 Android 进行了测试，测试平台包括 Android 2.3.3 模拟器和智能电视机顶盒。基于模拟器和智能电视平台，本文对表 3 中所有包装格式和编解码标准的本地文件和 http 网络文件进行了全面测试，测试全部通过，音视频能正常播放且保持同步。

表 3 Stagefright 扩展后 Android 支持的包装格式和音视频编码

Table3 Supported containers and A/V codec of new Android

	originally supported	newly supported
video container	mp4,3gp	mov,avi,mkv,flv,f4v,ts,rmvb,MPEG-PS,asf,wmv
audio container	mp3, m4a, ogg, wave	wma,aac,amr,ac3,mka(Matroska),mp2,ra(RealMedia),flac
video codec	AVC,XVID,MPEG4,H.263	microsoft MPEG-4,MJPEG, RealVideo4,MPEG2,MPEG1,WMV
audio codec	MP3,AMR,AAC,Vorbis,PCM	WMA,AC3,MP2,Cooker,FLAC

测试结果表明：扩展 Stagefright 后，Android 系统不仅支持原音视频包装格式和编码标准，还支持原 Android 不支持的所有常见格式，达到了 Android 多媒体框架扩展的目的，满足了智能电视对众多包装格式和编码标准的需求。此外，在三网融合背景下，数字电视节目通过 Cable 线广播的同时，也通过 rtsp 协议广播互联网网络电视节目，新 Android 不仅支持本地文件和 http 网络多媒体流，也支持 rtsp 实时流。部分视频测试实物图如图 5 所示。

4 结论

本文基于 FFmpeg 创建并植入了适配于 Android Stagefright 框架的解包装和解码插件，达到了扩展 Android 多媒体框架的目的。扩展后的多媒体框架支持所有常见文件包装格式和音视频编码标准，并同时支持本地文件、http 协议和 rtsp 协议下的多媒体流，满足了智能电视对众多包装格式、编码标准和信道的需求。相对于传统扩展方案，本方案未改变 Android 的 Java Framework，因此对应用程序的开发无任何影响，基于标准 Android SDK 开发的多媒体

Fig.5 Video test result

图 5 视频测试结果

播放器就能直接在扩展后的 Android 上运行，应用程序兼容性好。此外，扩展后的多媒体框架移植性好，仅需添加和替换若干动态库即可移植到模拟器、手机、数字电视机顶盒和平板电脑上。

参考文献:

[1] 贺冯良,张敏,马玲芳. 基于 Android+ARM 平台的车载信息系统的应用[J]. 信息与电子工程, 2012,10(3):271–273. (HE Fengliang,ZHANG Min,MA Lingfang. Application of vehicle information system based on Android and ARM platform[J]. Information and Electronic Engineering, 2012,10(3):271–273.)

[2] 尹显东,李在铭,姚军,等. 图像压缩标准研究的发展与前景[J]. 信息与电子工程, 2003,1(4):327–331. (YIN Xiandong, LI Zaiming,YAO Jun,et al. A Survey on Trends of Image Compression Coding Standards[J]. Information and Electronic engineering, 2003,1(4):327–331.)

[3] Barbosa A,Goncalves J,Ribeiro A N,et al. Integration of SIP protocol in Android Media Framework[C]// International Conference on Computer as a Tool(EUROCON) 2011 IEEE. Wuhan:IEEE Press, 2011:1–4.

[4] Schmidt D G,F Kuhns. An Overview of the Real-Time CORBA Specification[J]. Computer, 2000,33(6):56–63.

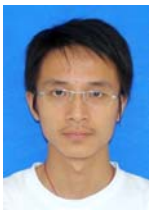
[5] 曾司凤,李凌. InTouch 基于 CORBA 的网络数据交换[J]. 信息与电子工程, 2004,2(1):10–11. (ZENG Sifeng,LI Ling. The Net Data Exchange of InTouch Based on CORBA[J]. Information and Electronic Engineering, 2004,2(1):10–11.)

[6] Tsai Chun-Shian,Chen Hsuan-Liang. The Implementation of Multimedia Decoder Framework for Android on PAC Duo Platform[C]// Digital Image Computing Techniques and Applications(DICTA) 2011 International Conference. Noosa,Queensland,Australia:IEEE Press, 2011:382–387.

[7] HE Jin,HE Jiaming. The Research of Plug-in Extension Technology Based on Android Multimedia Player Platform[C]// Computer Science and Service System(CSSS) 2011 International Conference. Nanjing:IEEE Press, 2011:874–877.

[8] SONG Maoqiang,XIONG Wenkuo,FU Xiangling. Research on Architecture of Multimedia and Its Design Based on Android[C]// Internet Technology and Applications,2010 International Conference. Wuhan:IEEE Press, 2010:1–4.

作者简介:



温　伟(1988–)，男，湖南省岳阳市人，在读硕士研究生，主要研究方向为多媒体通信与处理和智能电视系统.email:wenwei20202@126.com.

刘荣科(1973–)，男，西安市人，博士，博士生导师，主要研究方向为空天信息传输与处理、无线宽带多媒体通信、网络信息论与编码和专用集成电路设计.