基于 Android 系统的流媒体服务器①

张兴龙, 周渊平, 邓昌明

(四川大学 电子信息学院, 成都 610065)

摘 要: 为了解决手持移动设备之间的实时视频传输, 提出了基于 Android 系统的轻型流媒体视频传输系统的设计方案. 利用流媒体传输控制技术, 通过移植优化 live555 项目, 实现了基于 Android 系统的流媒体服务器. 主要实现了裁剪和移植 live555 和 FFmpeg 项目至 Android 系统, 优化解决了 live555 不支持 MP4、avi 等通用媒体文件的问题.

关键词: Android; 流媒体; live555; 视频传输; FFmpeg

Streaming Media Server Based on Android

ZHANG Xing-Long, ZHOU Yuan-Ping, DENG Chang-Ming

(College of Electronics and Information, Sichuan University, Chengdu 610065, China)

Abstract: Present a streaming media server application based on Android design for solving the real-time video transmission between the handheld mobile devices. The streaming media server is designed and implemented by streaming media technology and control protocol technology and expanding open-source multimedia server live555 project. Port live555 and FFmpeg to Android system, solve the problem that live555 does not support MP4, avi and other common media files.

Key words: Android; Streaming Media; live555; video transmission; FFmpeg

随着移动通信技术和多媒体技术的迅速发展, 移动流媒体服务有着很大的市场潜力. 特别是随着手持移动设备性能的大幅提升, 基于 Android 系统的应用越来越受到市场的青睐[1]. 本文结合 VLC 和 live555 项目设计出基于 Android 系统的视频流媒体播放系统. 其中, VLC 是一款开源的支持多种媒体格式的跨平台多媒体播放器及框架, live555 是一种解决流媒体方案的开源项目. 通过流媒体控制传输, 软硬件解码, 在局域网内实现两台 Android 设备之间流畅传输播放多格式视频文件的功能. 文章具体讨论分析系统服务器端软件的设计与实现, 考虑到现有的 live555 工程只支持基本流操作和部分媒体文件, 通过移植、优化 live555 开源工程, 加入文件解析模块, 设计和实现了能够支持 MP4, avi 等通用文件格式的流媒体传输系统.

1 系统的总体架构

系统主要应用于基于 Android 系统的手持设备, 分为服务器端和客户端. 服务器主要负责定位客户端请求文件, 建立网络间通信连接, 解析客户端请求的视频文件, 分离出基本的音视频流进行 RTP 打包, 将音视频流实时传输到客户端. 客户端通过修改 Android 内核多媒体处理相应模块, 在接收到数据后, 首先匹配数据格式, 进行解复用, 同步音视频数据, 软硬件解码, 完成实时播放.

服务器端界面开发[2]比较简单, 只需要基本的界面通过 JNI 调用底层 C 语言的主函数. 在主函数中首先需要建立任务调度器, 然后创建 RTSP 服务器. 服务器创建 Socket 并在 TCP 的 554 端口进行监听, 同时把连接处理函数句柄和 Socket 句柄关联起来传给任务调度器. 主程序进入任务调度的主循环中等待客户端请

① 收稿时间:2013-05-06;收到修改稿时间:2013-05-27

求连接. 当客户端请求连接服务器时, select 函数返回对应 Socket, 系统调用相应处理函数句柄创建起与客户端对话, 即与客户端建立起 RTSP 连接. 如有客户端向服务器发送请求, 服务器从客户端 Socket 读取数据, 对请求数据进行转换, 分离出指令并做相应的处理(包括 describe,setup,play 等). 处理完毕后将 RequestBuffer 进行重置, 为下一次请求做好准备. 如此循环执行, 以实现流媒体数据的传输和播放.

2 服务器端软件实现

2.1 FFmpeg 到 Android 平台的移植

FFmpeg 是一个功能强大的具有文件解析, 音/视频编解码等功能的开源解决方案. 服务器需要利用 ffmpeg 相应的库对现有的封装文件进行解析, 所以需要将 FFmpeg 库移植到 Android 系统中. 为了提升代码质量, 系统只选择了将服务器端所需要的解封装文件部分和客户端所需的解码部分编译到链接库. 配置相应参数后, 遴选出编译时所需的源文件移植到 Android 系统中, 降低代码冗余.

在 Android 系统下, Android.mk 文件语法和 linux 系统中的 makefile 文件语法格式不同[3], 因此需要编写新的 Android.mk 文件. 在 FFmpeg 中, avutil 是基础模块, avcodec 模块的编译基于已经编译好的 avutil 模块, avformat 基于前两者. 所以, 编译移植的顺序为 avutil、avcoedec、avformat, 根据 Android.mk 的格式, 为每个模块引入体系架构, 链接库, 头文件以及所需要的源文件等内容, 完成后通过 ndk-build 得到 Android 系统所支持的库文件[4]. 因为 Android.mk 文件内容很长, 在此不在详述.

2.2 live555 到 android 平台的移植

live555 是一个为流媒体提供解决方案的开源项目, 它实现了对标准流媒体传送协议的支持[5]. 系统服务器端需要在 Android 上的扩展 live555 项目, 所以, 首先需要将库移植到 Android 上. 具体移植过程如下:

① 新建一个 Android 工程, 将源码文件放到工程的 jni 目录下: jni/live555.

② 创建 jni/Android.mk, 将需要编译的源文件加入到 LOCAL_SRC_FILES 变量, 将头文件加入到 LOCAL_C_INCLUDES, 并加入参数 LOCAL_CPPFLAGS+= -fexceptions.

③ 新建并编写 jni/Application.mk 文件. 特别地, 需要加入: APP_STL := gnustl_shared

④ 运行 ndk-build 得到 live555.so 文件.

2.3 服务器软件构建

将需要的库文件移植到 Android 系统后, 下一步, 需要搭建服务器端程序. live555 只支持一些基本流操作. 因为需要对 live 代码进行扩展, 以期实现对 Android 平台下常用的 MP4, avi 等音视频格式的兼容. 文章利用 FFmpeg 库中的解析模块来扩展 live555 所支持的文件格式.

(1) 首先, 定义 RTSPServer 类 AnroidRTSPServer, 继承自 DynamicRTSPServer. 该类重新实现 LookupServerMediaSession 和 CreatNewSMS 函数. LookupServerMediaSession 函数通过调用 DynamicRTSPServer::LookupServerMediaSession 来处理 live555 支持的基本流情况, 如果是 MP4 等音视频复合文件, 则调用自身的 CreatNewSMS 函数创建一个新的服务类, 通过服务类创建相关基本流的子会话对象.

(2) 定义以下几个类处理解析基本流

AndriodDemux 继承自 Medium, 该类实现了 ffmpeg 中相关函数来完成音视频复合文件的解析工作, 分离出文件中基本媒体流的数据.

AndriodDemuxedElementaryStream 继承自 FramedSource 类, 可以将该类看成一个虚拟 source 完成与 RTPsink 的交互, 类通过调用 AndriodDemux 类实例获取所需的基本流数据.

AndriodServerDemux 继承自 Medium, 该类是个服务类, 主要用来创建 AndriodDemux 及对应的具体 Mediasubsession 实例.

类的层次结构如图 1 所示.

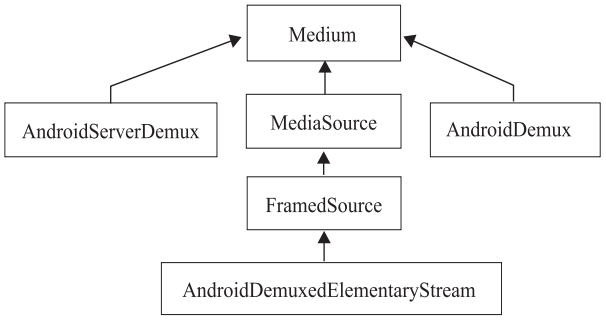

图 1 扩展类的层次结构

(3) Mediasubsession 类的实现

live555 中每种类型媒体流都有自己的 subsession

实现, 通过 createNewStreamSource 函数来创建自己的 source. 通常 MP4 和 avi 格式的视频文件主要包含 h.246,mpeg4,mp3,aac 等媒体格式, 所以, 我们重新实现了各个子类的 Mediasubsession:

AndroidH264ServerMediaSubsession 类, 继承自 H264VideoFileServerMediaSubsession, 实现将 packet 中获取的 h264 的 es 流, 交给 H264VideoStreamFramer 作后续处理.

AndroidMPEG4ServerMediaSubsession 类, 继承于 MPEG4VideoFileServerMediaSubsession, 这里需要注意的是, 从 packet 中获取到的数据并不是严格的 ES 流. Live555 中与处理 mpeg4 es 流相关的类有两个, MPEG4VideoStreamFramer 与 MPEG4VideoStream DiscreteFramer, MPEGVideoStreamFramer 只能处理严格的 MPEG4 ES 流. MPEG4VideoStreamDiscreteFramer 继承自 MPEG4VideoStreamFramer, 它可以处理 VOS, 也可以处理一个个的 BOV 及 VOP, 这里选择 MPEG4VideoStreamDiscreteFramer 作后续处理.

mp3 格式子会话 MP3ServerMediaSubsession, 继承于类 MP3AudioFileServerMediaSubsession.

aac 格式子会话 aacServerMediaSubsession, 继承于类 FileServerMediaSubsession. 主要实现了 create NewStreamSource 及 createNewRTPSink 函数.

通过以上类的扩展, 当客户端请求播放 avi 或者是 MP4 文件时, 首先创建 ServerMediaSession 类处理会话描述, 然后实例化服务类 AndriodServerDemux, 并调用类的函数创建音视频子会话描述类 ServerMediaSubsession. 以 H264 格式为例. 创建 AndroidH264ServerMediaSubsession 类. 在请求播放的阶段, 系统首先调用子会话的 startStream(), 函数内部调用 MediaSink::startPlaying(), 然后通过特定媒体的 sink 子类实现 RTP 的打包与发送[6]. 特别地, 在往 RTP 包中填充数据时, 如果 buffer 中不存在数据, 需要调用 Source 上的 getNextFrame 函数获取数据, 系统在此处做了扩充, 对于 MP4 格式文件, 不再是直接读取文件获取, 而是调用 AndriodDemuxedElementaryStream 类中的 doGetNextFrame() 函数实现, 函数会调用 AndroidDemux::doGetNextFrame() 函数来解析所需要的基本流数据, 并将获取的数据压入到 buffer 中, 然后通过相应的 sink 子类进行 RTP 打包以及分片操作, 最后发送到客户端. 数据处理过程如图 2 所示.

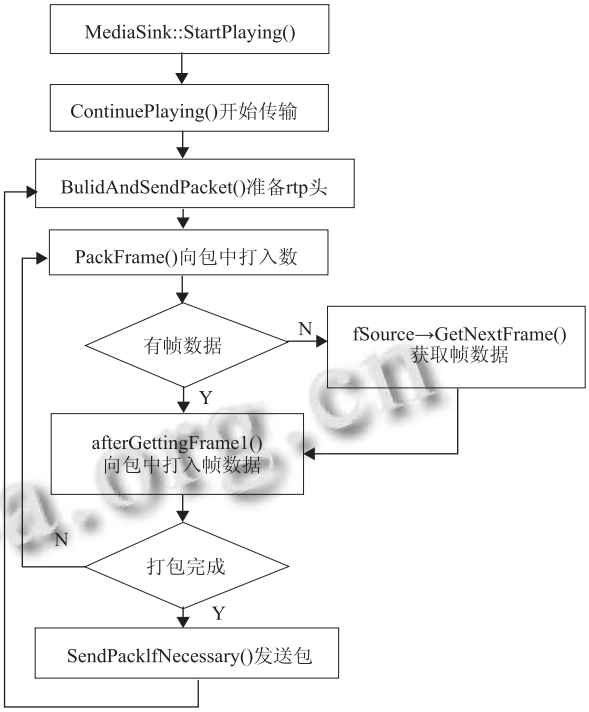

图 2　数据的打包与发送过程

(4) 处理 avi、mp4 文件的一些注意事项

通过 FFmpeg 获取的数据一般保存在 packet 中. 对于 avi 文件, 直接提取出 packet 数据即可. ?对于 MP4 文件, 需要作特殊处理: 提取 h264 数据时, SPS 及 PPS 信息保存在了 AVCodecContext 的 extradata 数据域中, 所以需要通过流过滤器"h264_mp4toannexb"获取数据. 另外 packet 中的数据也不是标准 nalu 单元, 需要将前 4 个字节替成开始符. 提取 mpeg4 数据时, vos(0x000001B0) 也是存储在 AVCodecContext 的 extradata 数据域中, 需要将 extradata 中的数据添加到流的开始处[7].

3　系统测试

开启流媒体服务器端软件, 服务器打开监听程序, 监听客户端请求. 这里利用了 VLC 软件作为客户端来测试, 点击 VLC 菜单文件, 打开网络串流, 选择 RTSP, 输入请求服务器的 IP 地址, 请求的媒体文件以及端口号. 点击确定. 实时视频播放画面图像质量高, 播放效果比较理想. 测试结果如图 3、图 4 所示.

测试表明, 在 Android 移动系统下, 对于 MP4,avi 等视频文件, 服务器对客户端的视频传送图像质量高, 稳定可靠.

图 3 系统测试设置

图 4 系统测试效果

4 结束语

当今，智能手机和平板电脑越来越受到人们的欢迎. 手机间的数据传输也越来越频繁，当两部手持设备需要共享视频文件的时候，往往不想对视频文件进行先传输再播放，因此，基于流媒体的实时视频播放系统会有一定的市场需求，本系统实现了这种需求，具有一定的实际意义.

参考文献

1 焦铬,易小波,李仁发.基于嵌入式 Internet 的远程视频监控系统设计.计算机技术与发展,2009,19(5):176–179.
2 王作成,摆玉龙,李昂.基于 Android 平台的手机视频优化编码.计算机系统应用,2013,22(1):126–128.
3 杨丰盛.Android 应用开发揭秘.北京:机械工业出版社.2011.
4 王有禄,李代平.Android 系统下基于 NDK 方式的图形开发.计算机系统应用,2012,21(12):56–59.
5 李校林,刘海波等.RTP/RTCP,RTSP 在无线视频监控系统的设计与实现.电视技术,2011,35(19):89–92.
6 茅炎菲,黄忠东.基于 RTSP 协议网络监控系统的研究与实现.计算机工程与设计,2011,32(7):2523–2530.
7 曾金,毛燕琴,沈苏彬.嵌入式流媒体服务器的设计和实现.计算机技术与发展,2011,21(7):81–84.