Android机顶盒媒体播放器框架

# 概述

目前我们的媒体框架支持本地播放和网络播放，功能方面主要包括：

1. IPTV环境的直播播放；
2. IPTV环境的点播播放；
3. 根据用户指定的URL实现本地播放；
4. OTT环境的网络播放；

# 播放器设计

## 播放器概述

播放器主要功能是实现对多媒体音视频的解复用、解码到播放的过程。它基于ffmpeg源码实现，媒体流首先由ffmpeg进行解复用，得到音频、视频和字幕的ES流。根据头信息匹配对应的解码器，由于硬件不支持音频和字幕的解码，所以音频和字幕只能使用ffmpeg自带的软件解码器，而视频根据编码格式优先选用硬件解码器，若硬件解码器不支持此格式，则会自动选用ffmpeg的软件解码器进行处理。

目前我们芯片硬件支持视频解码格式有H264、MPEG2、MPEG4、VC-1、AVS、RV、VP8、VP6等格式，Video IP由Hantro厂商提供。Hantro厂商的硬件解码器提供的api接口被封装为ffmpeg中的AVCodec解码器对象，硬件解码器与ffmpeg的其他软件解码器公用一套api接口，对上层屏蔽了不同解码格式以及软件解码和硬件解码的差异。在播放器初始化的时候将所有的解码器注册到FFMPEG的解码器列表中，供后续解码使用。

根据不同格式的播放，我们主要区分为3类播放流程：

1. IPTV播放：iptv分直播和点播，直播可以为单播或者组播方式，目前市场下发的都是ts流，走ts播放流程；点播为单播方式，当下发的流为ts流时走ts播放流程，下发的流为nts流时（一般为mp4格式），走nts播放流程。
2. OTT播放/本地播放：第三方APK播放（hls格式除外）或者本地播放走android标准媒体框架流程，封装ZTEPlayer播放器。
3. OTT中的hls播放：最初设计的ott播放流程使用ffmpeg解析码流，在解析m3u8（hls）格式时，起播速度较慢，并且由于对ffmpeg掌握程度不够，遇到问题比较难解决，后经讨论决定使用vod解析hls。 后经过改造在mediaplayer层创建ZPlayer播放器，通过playerAdapter适配vod，来播放ott中的hls。



图2-1 播放器总体流程

1. 图中第一路为IPTV播放流程：

ts流：VOD—>Decoder(Ts\_player)—>CTC—>SDK

Nts流：VOD—>Decoder(nts\_player)—>SDK

1. 第二路为OTT/本地播放流程：Mediaplayer—>ZTEPlayer—>SDK
2. 第三路为OTT的HLS播放流程：

Mediaplayer—>ZPlayer—>Adapter—>VOD—>Decoder(tsplayer)—>CTC—>SDK

## IPTV播放

### 系统架构

在介绍IPTV播放流程之前先熟悉一下vod的框架结构，在机顶盒框架中，VOD主要是五层结构：控制层、媒体中间件层、媒体预处理层、媒体播放器控制层、平台隔离层。

**控制层：** 该层主要负责接收外部程序发来的控制命令和分派处理这些命令，并且根据需要反馈给外部程序应答消息。msgpump模块负责接收和发送消息，mp\_evtprocessor模块主要将msgpump接收到的数据进行分派和处理，Evtnotification用于向外部程序反馈应答消息或者发送一些状态消息，osd显示控制和非媒体播放控制。

**媒体中间件层：** 包含ZTESC模块或者第三方流媒体中间件，主要负责与媒体服务器建立通讯，目前ZTESC支持rtsp、iudp、http、rtp和mms。

**媒体预处理层：**封装对中间件的调用，尤其在支持第三方中间件时作为一个适配层。该层增加对媒体数据的保护处理，如QOS模块和FEC模块。evtprocessor收到关于媒体播放的控制命令都需要由StreamClient在中间件和播放器进行适配，中间件负责与流媒体服务器交互，播放器负责与解码器交互。该层还有一个重要的作用就是媒体预处理，即对从流媒体服务器接收到数据进行FEC处理、丢包重传处理、QOS处理后进行网络统计，数据最后经过DRM解密处理后送给Media\_player 。

**媒体播放器控制层：**media player、media player panel和其他扩展的播放器模块，负责对解码器的控制，media player负责和节目播放相关的控制，例如打开并且启动解码器、时移操作、关闭解码器等，media player panel负责设置解码器的音量、静音操作、声道切换等。

**平台隔离层：** 与硬件相关的适配层，包括对解码器和osd显示的控制，向上层应用提供的接口统一，由平台api作支撑，即Decoder的HAL\_XXX接口，对于个别平台差异的处理放在porting模块来适配。

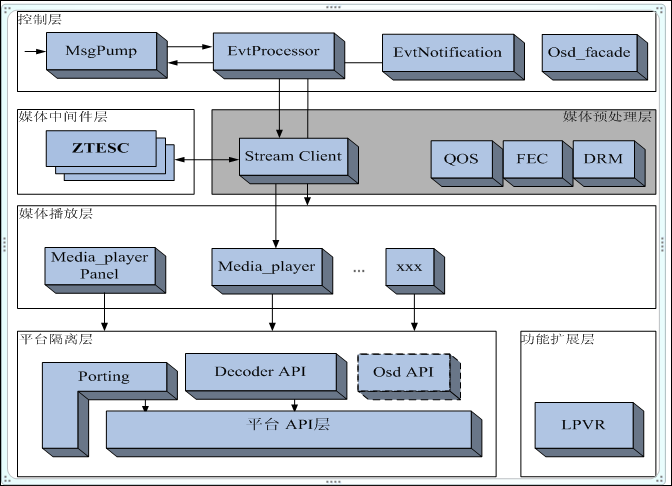


图2-2 VOD框架简介

 图2-3 sdk系统模块位置

组件的说明参见表1.1

表1.1

|  |  |
| --- | --- |
| 组件标识 | 组件功能概述 |
| HAL层 | HAL层主要是通过消息处理模块发消息与调用相应解码接口来完成调用解码器接口相应功能，向上层应用提供解码器相应功能，并设置解码器的相关状态。可以说HAL层是上层应用和解码器的桥梁 |
| 业务数据存储模块 | 负责媒体流数据的缓存，提供写数据接口给HAL层，提供读数据接口给解复用模块 |
| 消息处理模块 | 负责接收HAL层发下来的消息放到消息队列里，同时提供读取消息的接口给主控模块调用 |
| 主控模块 | 负责状态机的运行与消息处理，根据调度策略对各个子模块进行调度 |
| 解复用模块 | 负责对原始码流的解复用工作，把解复用的数据放到待解码队列里供解决模块使用 |
| 解码模块 | 从各个待解码队列里取数据进行解码 |
| 后处理模块 | 从解码模块获取已经解码好的数据进行后处理 |
| 同步模块 | 对已经完成后处理的数据进行同步策略的处理 |
| 输出模块 | 调用硬件驱动将待输出的数据进行输出 |
| HDMI自适应模块 | 负责检测内核上报的HDMI热插拔状态，完成EPG界面的显示切换，并发消息给主控模块，让主控完成视频输出的切换 |
| 混音模块 | 负责两路音频的混音 |
| 音频透传模块 | 负责DD/DD+、DTS等音频的透传 |

SDK系统中包含业务数据存储模块、消息处理模块、主控模块、解复用模块、解码模块、后处理模块、同步模块、输出模块、HDMI自适应模块、混音模块和音频透传模块。从应用层下来的有两类数据，一类是控制消息，一类是媒体流数据。控制消息从接口层发送到消息处理模块，然后主控模块主动从消息处理模块取消息进行处理，在处理消息的过程中可能会进行状态的切换，主控通过调用解复用模块、解码模块、后处理模块、同步模块、输出模块的消息处理函数和状态处理函数来达到调度的目的，使整个系统运转起来。

视频解码模块在播放器初始化时，注册自己支持的硬件解码器，每个解码器对应了一个在avcodec.h中定义的AVCodecID号。当媒体流下发以后，首先在ffmpeg中探测视频格式，判断码流类型，先根据AVCodecID匹配已注册过的硬件解码器，如果在解码器表中，就使用硬件解码功能。否则就使用FFMPEG软件解码功能。视频ES数据解码完经过同步处理以后，得到的YUV数据送入显示单元进行显示。其视频解码软件流程如图2-4所示。



图2-4 视频解码软件流程

音频解码模块直接使用FFMPEG的软件解码功能，然后通过audioflinger将声音数据送到底层PCM音频设备中进行输出。

同步处理模块主要是通过同步算法，实现音视频和字幕的同步输出。这里主要是通过音视频和字幕的时间戳进行同步，以音频时间戳为基准，视频和字幕同步到音频。

显示单元其实就是OSD模块，主要将RGB数据和YUV数据分别送到图形层和视频层进行显示。图形层主要是采用framebuffer框架输出，视频层主要采用V4L2框架输出，音频输出单元主要将解码后的数据通过alsa输出到音频设备上。

### 播放流程

 图2-5 VOD与Decoder运行视图

1. 在播放过程中VOD的Media\_player会按下面的顺序调用HAL接口：
2. HAL\_DecInit－>HAL\_DecOpen－>其他HAL\_Decxxx函数
3. 其中HAL\_DecInit主要针对平台差异的处理放在porting模块来适配，主要获取机顶盒相关配置进行一些初始化，并创建sdk相关线程，如解码线程、后处理线程、同步输出线程等；
4. Decoder Agent调用HAL\_DecOpen注册各HAL\_Decxxx函数接口；
5. Decoder Agent调用HAL\_DecData，向解码器发送数据(ts流)；
6. 通过HAL\_Decxxx函数接口进行播放、暂停、快进、快退等各种播控操作；
7. VOD的Media\_player在主循环内部，将数据发送解码器，并周期性调用HAL\_DecProcess，用于处理和监控解码器的工作过程，反馈解码器工作过程的各种状态。

 图2-6 播放初始化流程



图2-7 播控调度流程

### 数据流

IPTV中ts流和nts流播放的主要区别：

ts流播放一般为iptv直播播放场景，由vod进行iptv协议解析得到（单播或组播协议），ts流输出后由ffmepg进行解复用得到独立的音频和视频es流。

nts流播放一般为iptv点播播放场景（如MP4格式），由ffmpeg解析http协议直接从服务器获取码流，然后ffmpeg进行解复用得到独立的音频和视频es流。

Ts和nts播放流程，是把sdk挂在vod线程下，编译生成ZTEPlayer。

 图2.8 iptv播放的 ts/nts数据流处理

Sdk中主要处理音视频流的线程包括：

1. 播放器主线程(player\_thread)：运行状态机，处理播控消息，控制整个播放流程；
2. 视频解码线程(decode\_video\_thread)：从解复用后的队列中获取包数据送与解码器解码（通过ffmpeg的avcodec\_decode\_video2 接口调用硬件解码器，解码器注册过程将后面介绍），得到AVFrame结构的帧数据；
3. 音频解码线程（decode\_audio\_thread）：调用ffmepg的avcodec\_decode\_audio软件解码，完成音频数据的解码。
4. 视频后处理线程（ppu\_proc\_thread）：其中ppu\_proc\_aux\_thread用于处理软件dit。
5. 同步输出线程(sync\_output\_proc\_thread)：进行音视频同步后进行音视频输出。

在DecStart中解复用初始化时，得到播放文件的基本信息中，如果包含视频信息，则需要创建视频解码播放线程。视频频解码播放线程创建后，先从解复用后的视频数据队列中取得视频数据，然后调用ffmepg的avcodec\_decode\_video解码，完成视频数据的解码，然后通过ppu后处理，最终通过同步输出线程将YUV图像送显。视频解码播放流程如图2-9所示。



图2-9 视频数据流

解复用初始化时，得到播放文件的基本信息，如果包含音频信息，则需要创建音频解码播放线程。ffmpeg为音频解码数据申请存放空间，大小为192000bit，为1s钟的48kHz 32bit数据大小，接下来进入循环，每次从音频数据队列中取得音频数据包，调用ffmpeg的avcodec\_decode\_audio软件解码，完成音频数据的解码，最终通过AudioTrack将解码后的声音数据输出。音频解码输出流程如图2-10所示。

 图2-10 音频解码输出数据流

adec\_buff[MAX\_DECODER\_NUM][MAX\_AUDIO\_FRAME\_SIZE]作为音频解码模块与音频后处理模块音频数据之间的桥梁。MAX\_AUDIO\_FRAME\_SIZE为2M大小。用一个rdx\_offset[]读偏移和frame\_numb帧大小来操作此buffer的数据，若偏移超过2M，则从头开始计算偏移。如下图2-11所示，1代表这次数据的的读偏移，黄色区域代表帧大小，2代表下次数据的读偏移。

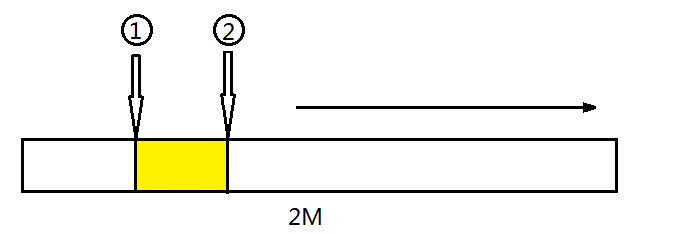


图2-11 adec\_buff操作示例

## OTT播放/本地播放

### ZTEPlayer系统架构

ZTEPlayer播放器实现框架如图2-12所示, 在getPlayerType中创建一个新的播放器ZTEPlayer

与Stagefrightplayer并列，红色部分是我们需要实现的。



图2-12 基于ZTEPlayer播放器实现框架

媒体文件主要通过android上层mediaplayer类的setDataSource方法，将媒体文件路径通过参数传给ZtePlayer，再通过调用已经封装好的ffmpeg接口，完成对该媒体文件的解复用。解复用后就得到对应的音视频ES流和字幕ES流，为解码单元提供原始数据流。



图2-13 ZtePlayer创建线程过程

PrepareAsync功能主要在Zteplayer类实现，这个接口主要创建播放主线程，解复用初始化线程，音频解码线程，视频解码线程，ppu后处理线程，同步输出线程以及hdmi热插拔线程。

解复用初始化独立成线程主要是为了解决果果乐园apk进去后会播放小视频，此时播放处于prepareAsync状态下，运行解复用初始化会耗较长时间，此时apk无法移动焦点，因此把解复用初始化独立成线程，非阻塞方式下焦点可以很快获得。

### ZTEPlayer播放流程

一个简单的应用播放器必需要调用的接口如图2-14所示：



图2-14 应用播放器基本调用的接口

Mp是Java应用层的播放器类，按顺序调用上面三个播放接口即可实现简单的视频播放功能。

首先看一下设置媒体源文件路径软件调用流程，如图2-15所示。首先Java层mediaplayer设置媒体源文件路径，通过JNI调用android framework层C++代码实现，然后通过调用本地框架的客户端mediaplayer接口，再通过binder机制调用到服务端mediaplayerservice接口，最后调用到我们zteplayer接口，最终是在zteplayer的接口setDataSource中得以实现。这个函数接口主要功能是记住媒体文件路径，方便后面的接口使用这个路径。



图2-15 设置媒体流文件路径软件流程

其次是下面的prepareAsync接口实现，具体软件流程如图2-16所示：



图2-16 prepareAsync软件流程图

前面两个接口调用完成后，接下来就可以调用start开始播放的接口，具体软件调用流程如图2-17所示：



图2-17 start软件流程图

### ZTEPlayer数据流

 图2.18 OTT/本地播放的数据流处理

本地播放通常从本地读取fd文件方式获取码流，ott播放一般下发http格式的url地址，由ffmepg解析http协议从服务器获取码流。解复用以及后面的音频和视频解码输出流程同IPTV播放方式。

本地播放和ott播放流程，都是apk直接调用android的mediaplay媒体框架，mediaplay再调用ZTEPlayer播放器。libmediaplayerservice目录下的mediaplayer和zteplayer编译生成mediaserver。

## OTT中的HLS播放

为了保证VOD模块的独立性，在Android媒体框架与VOD模块之间通过ZTEPlayerAdater模块来进行适配，VOD模块通过ZTEPlayerAdater挂载在Android媒体框架之下。

### ZPlayer系统架构



图2-19基于ZPlayer播放器实现框架

### ZPlayer播放流程

由于ZPlayer和ZTEPlayer都走android标准媒体框架流程,上层播放过程基本一致

一个简单的应用播放器必需要调用的接口如图2-20所示：



图2-20 应用播放器基本调用的接口

ZPlayer与ZTEPlayer不同之处在于，它没有直接与sdk交互，而是通过playAdapter适配vod

( 通过MCSP\_InvokeService服务），由vod跟sdk进行交互，后面的流程跟iptv的ts播放流程一致。

### ZPlayer数据流

 图2.21 OTT 中hls播放的数据流处理

OTT中的HLS播放一般下发http格式包含m3u8后缀的url地址，由VOD解析http协议从服务器获取码流，根据hls协议解析出ts流，剩下的流程跟iptv的ts播放数据流程一样。

# 760EV3播放器代码分析

## IPTV播放流程代码解析

Iptv的ts和nts播放流程图如下

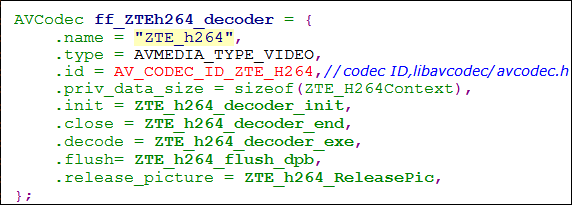


其中sdk底层硬件解码器通过avcodec\_register接口向ffmpeg注册，过程如下：

dec\_init—>ffmpeg\_global\_init—>ffmpeg\_register—>av\_register\_hwdecoder



以h264解码器为例，定义如下播放器接口：ZTE\_h264\_decoder\_exe最终调用硬件解码器提供的api接口进行解码。



## ZTEPlayer播放流程代码解析

JAVA层mediaplayer是具体创建播放器实例，其对应的代码如下：

Frameworks/base/media/java/android/media/MediaPlayer.java

Libmedia\_jni.so库主要实现JAVA和C++代码的对接，即JAVA层代码通过这一层封装可以直接调用框架层C++实现的代码。其对应的代码如下：

Frameworks/base/media/jni/android\_media\_MediaPlayer.cpp

本地媒体框架libmedia.so位于核心的位置，它对上层的提供的接口主要是MediaPlayer类，以及通过binder机制与媒体服务mediaplayerserver进行通信，实现具体的播放功能。其对应代码如下：

Frameworks/av/media/libmedia/MediaPlayer.cpp

媒体服务mediaplayerserver是Media的服务器，它通过继承libmedia.so的类实现服务器的功能。它根据playertype创建对应的播放器，这里出现了两种播放器类型，stagefrightplayer是android原生态默认的播放器，ZtePlayer是我们自己实现的一种播放器，所以代码里创建的是ZtePlayer播放器，所以mediaplayerservice的真正功能通过调用ZtePlayer播放器来完成。其对应的代码如下：

Frameworks/av/media/libmediaplayerservice/zteplayer.cpp

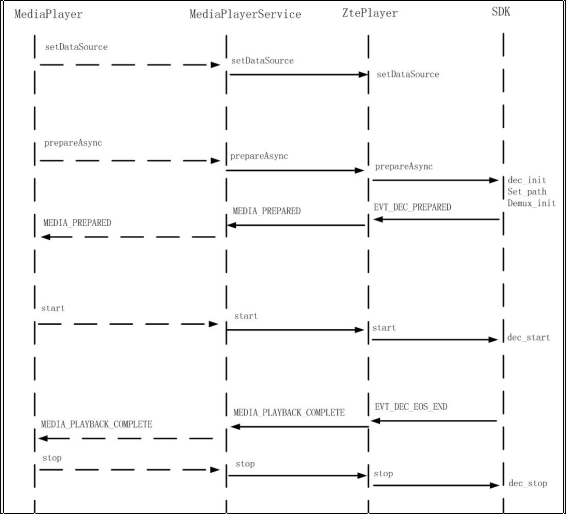


图2-22 ZTEPlayer与MediaPlayer和SDK主要交互过程

## ZPlayer播放流程代码解析

媒体服务mediaplayerserver层与ZtePlayer播放流程一致，功能实现通过调用ZPlayer播放

器来完成。其对应的代码如下：

Frameworks/av/media/libmediaplayerservice/Zplayer.cpp

Zplayer之下调用playAdapter，playAdapter通过MCSP\_InvokeService与vod通信，vod通过ctc接口与sdk通信，它们之间的基本交互过程见下图：

 图2-23 ZPlayer与MediaPlayer和sdk基本交互过程

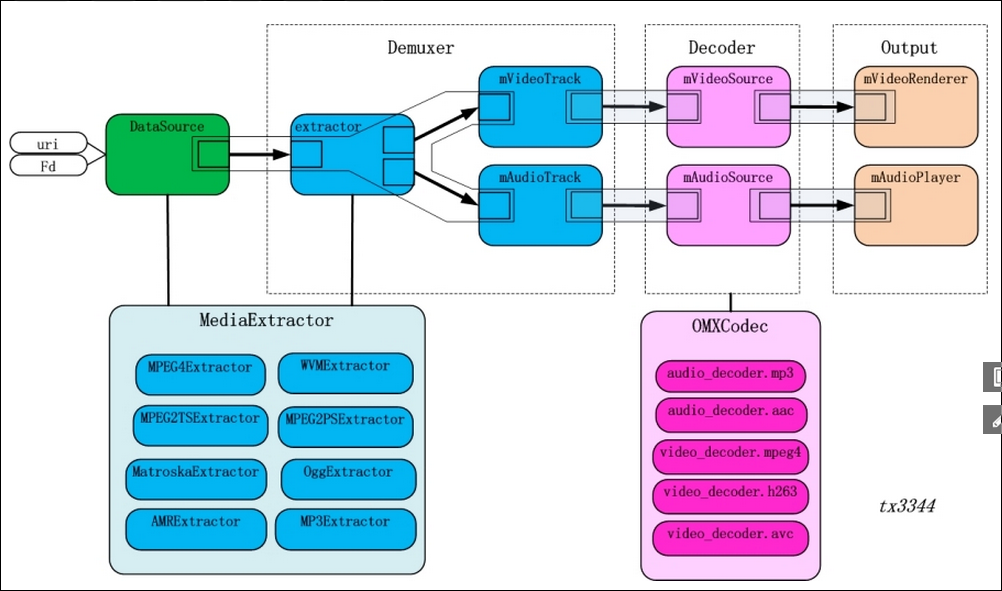
# 统一平台规划思路



第一步规划：iptv播放流程不变。OTT/本地播放和HLS播放合并，使用ZTEPlayer播放



第二步规划：iptv播放流程不变。OTT/本地播放和HLS播放，使用Stagefrihgtplayer和omx方式



第三步规划：使用MdediaExtractor封装ffmpeg解复用，与omx解码，进行联调

第四步规划：iptv和ott统一播放器，方案待定