Android多媒体开发技术

何俊林

自我介绍

- 曾就职于爱奇艺，主要参与Android播放器业务和需求开发，以及TV新播放内核开发和维护。

Agenda

- 流媒体传输协议
- 直播技术
- 音视频通话
- 音视频引擎构成
- 案例分享
- 常见播放方案

思考一些问题

- 音视频数据是怎么在互联网上传输的?
- 不同协议的使用场景是什么?
- 应用开发的，如何明确协议界限划分?

Agenda

- **流媒体传输协议**
- 直播技术
- 音视频通话
- 音视频引擎构成
- 案例分享
- 常见播放方案

流媒体传输协议

- RTP
- RTCP
- RTSP
- RTMP
- HLS
- HTTP-FLV

流媒体传输协议

- RTP
  - RTP全称（Real-time Transport Protocol），表示实时**传输协议**。主要用于**传递**音频和视频的标准数据包格式。RTP是基于**UDP协议**的，RTP服务器会通过UDP协议，每次会发送一个RTP packet。客户端通过解析RTP packet，读取其中的数据然后进行播放。
  - RTP packet的结构如下：
    - RTP Header：RTP 包的头部
    - contributing sources：个数为0-n个，所以可以为空。
    - RTP payload：即RTP要传输的数据

流媒体传输协议

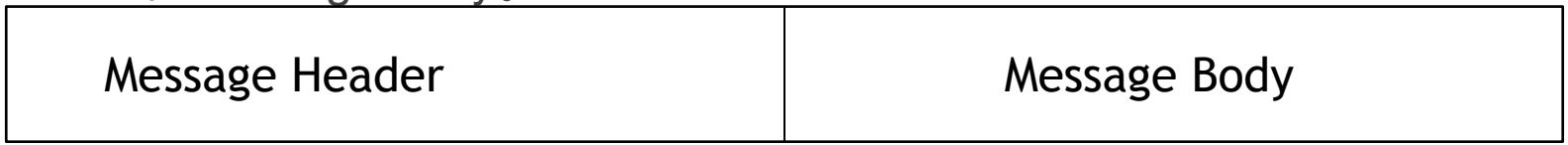
- RTCP
  - RTCP全称（Real-time Transport Control Protocol），实时传输**控制协议**，**不是传输数据**，同RTP一起用于**数据传输的控制、同步**功能。
  - 用于打包和发送RTP
  - 为RTP所提供的服务质量（Quality of Service）提供反馈，例如：传输字节数，传输分组数，丢失分组数，jitter，单向和双向网络延迟等等。

流媒体传输协议

- RTSP
  - RTSP全称（Real Time Streaming Protocol），实时流协议
  - 主要用来做流媒体服务器的**远程控制**，但它本身并不传输数据，是应用层协议，而是必须依赖于下层传输协议所提供的某些服务。RTSP可以对流媒体提供诸如播放、暂停、快进等命令操作。
  - 可以同时控制多个串流会话
- 一句话总结
  - RTSP负责**发起和终结**会话，RTP负责会话**数据的传输**，RTCP配合RTP做控制**数据包发送和反馈**

流媒体传输协议

- RTMP
  - RTMP全称（ Real Time Message Protocol），实时消息传输协议，RTMP协议是应用层协议，是要靠底层可靠的传输层协议（通常是TCP）来保证信息传输的可靠性的。
  - RTMP协议中的数据被称为Message（消息）。一个Message包含Message header和Message body。



流媒体传输协议

- RTMP
  - 在网络传输的时候，如果一个Message太大，那么它将会被**分段传输**。一个分段被称为Chunk，Chunk包含Chunk Header 和Chunk Body。

Chunk Header	Chunk Body

  - Chunk的长度初始长度固定为128个字节，但是这个值可以在建立连接之后进行修改。 其中Chunk Header包含Basic header 、Chunk Msg Header、Extend TimeStamp。具体的格式如下：

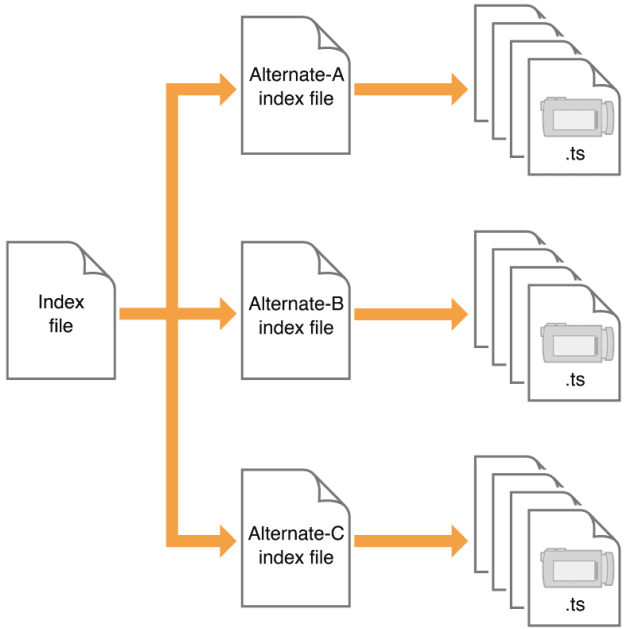
Basic Header	Chunk Msg Header	Extend TimeStamp	Data

流媒体传输协议

- HLS
  - HLS全称（**HTTP Live Streaming**），由苹果公司提出的基于HTTP的流媒体网络传输协议。是苹果公司QuickTime X和iPhone软件系统的一部分。它的工作原理是把**整个流分成一个个小的基于HTTP的文件来下载**，每次只下载一些。当媒体流正在播放时，客户端可以选择从许多不同的备用源中以不同的速率下载同样的资源，允许流媒体会话适应不同的数据速率
  - 传输视频的封装格式是TS
  - 定义了用来**控制播放**的m3u8文件（文本文件）
  - m3u8分为一级索引文件和二级索引文件

流媒体传输协议

- HLS
  - m3u8一级索引和二级索引文件




```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1064000
1000kbps.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=564000
500kbps.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=282000
250kbps.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=2128000
2000kbps.m3u8
```

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-TARGETDURATION:10
#EXTINF:10,
2000kbps-00001.ts
#EXTINF:10,
2000kbps-00002.ts
#EXTINF:10,
#ZEN-TOTAL-DURATION:999.66667
#ZEN-AVERAGE-BANDWIDTH:2190954
#ZEN-MAXIMUM-BANDWIDTH:3536205
#EXT-X-ENDLIST
```

流媒体传输协议

- HTTP-FLV
  - HTTP-FLV协议是封装在HTTP协议之上的， 每一个音视频数据都被封装成了包含**时间戳信息头**的数据包。而当播放器拿到这些数据包解包的时候能够根据时间戳信息把这些音视频数据和之前到达的音视频数据连续起来播放。
  - 业界常见的是将直播流式数据虚拟成为一个无限大的FLV(FLASH VIDEO)文件，并通过HTTP协议进行传输。客户端仅发送一次HTTP GET请求，请求中携带需要访问的直播流名，服务器返回HTTP响应，不携带消息体内容长度直接发送无限长FLV文件内容，或者使用HTTP CHUNK模式将无限长FLV文件按分段模式发送。客户端获得HTTP消息体中的FLV内容时即可播放。
  - 如 ed的方式进行的传输的响应头

```
HTTP/1.1 200 OK
Server: openresty
Date: Wed, 21 Sep 2016 07:38:01 GMT
Content-Type: video/x-flv
Transfer-Encoding: chunked
Connection: close
Expires: Wed, 21 Sep 2016 07:38:00 GMT
Cache-Control: no-cache
```

Agenda

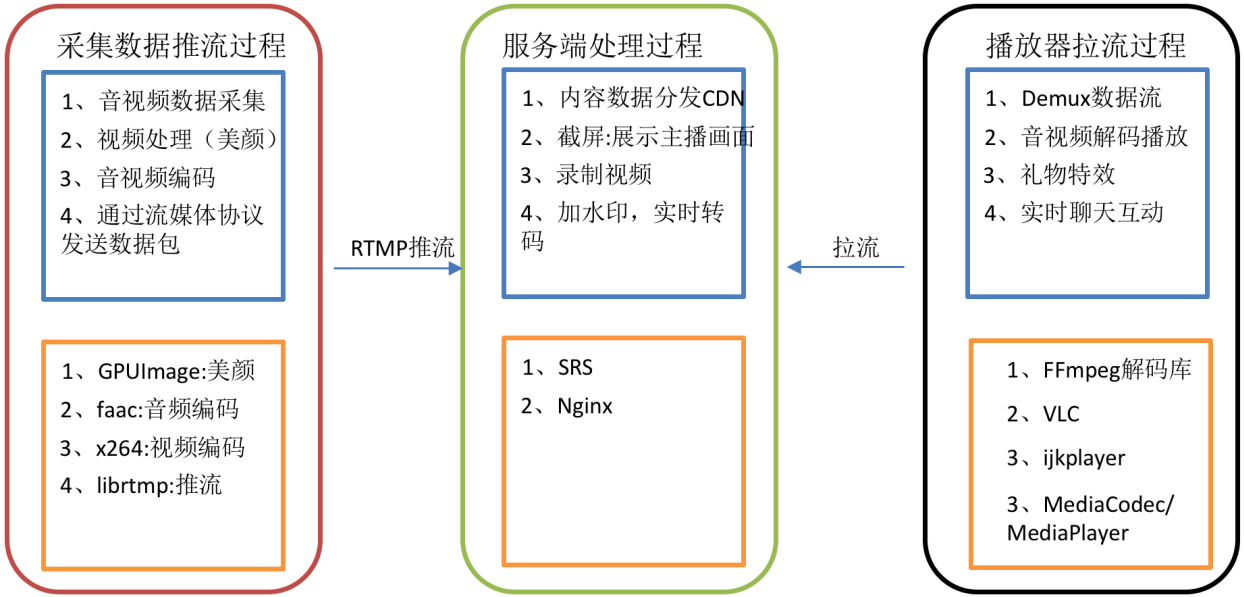
- 流媒体传输协议
- **直播技术**
- 音视频通话
- 音视频引擎构成
- 案例分享
- 常见播放方案

思考一些问题

- 你直播，我观看，中间经历了哪些过程?
- 斗鱼、映客、花椒切房间时，怎么能做到那么快直播?
- 网络抖动时，又想看直播，怎么办?
- 工作太忙，但又不想错过某明星演唱会直播，怎么回看?
- 网络质量差，怎么让主播和观众稳定使用?

直播技术

- APP直播流程



直播技术

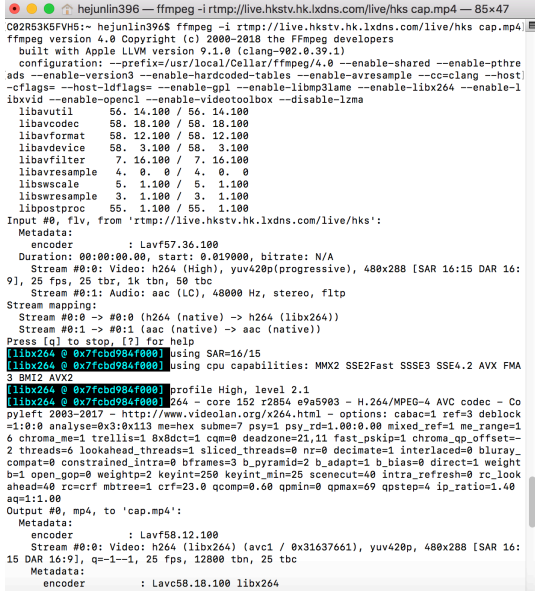
- 直播秒开分享
  - 下载到关键帧后立刻解码并渲染显示
  - 画面首帧不做音视频同步校验，首帧不缓存，直接渲染，后续帧再做音视频同步
  - 优化媒体信息解析速度（基于FFmpeg）
    - probesize
    - Analyzeduration
    - 减少这两个值，可以明显加快首首开，但是 probesize 过小，可能导致媒体信息解析错误
  - 用较低分辨率、码率起播，数据量少，可以很快渲染出首帧画面

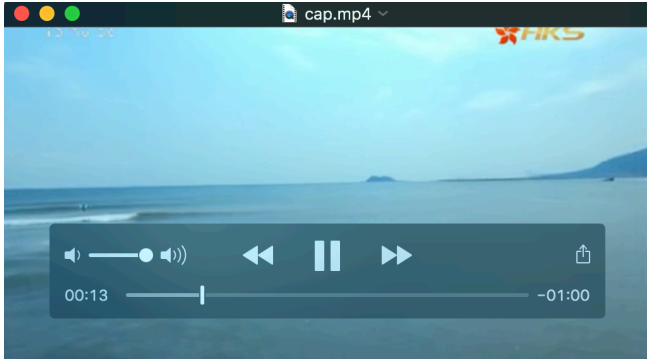
直播技术

- 动态码率自适应
  - 视频码率：就是数据传输时单位时间传送的数据位数，一般我们用的单位是kbps即千位每秒。
  - 视频分辨率：视频的宽和高像素值
  - 直播为了到达首帧秒开效果，通常用较低分辨率、码率起播，数据量少，可以很快渲染出首帧画面。如果首帧播放后，根据当前网络状况，切换到对应网速与相适应的分辨率、码率。
  - HLS、DASH协议可以支持进行动态码率自适应

直播技术

- 直播回看
  - 比如之前的世界杯，很多直播都是凌晨才能看，观众此时可能没法看直播，但是希望第二天能进行回看。需要通过FFmpeg对直播流进行转录到服务端。然后切片成ts，把ts索引放到m3u8文件中，给播放器进行播放，达到回看效果。





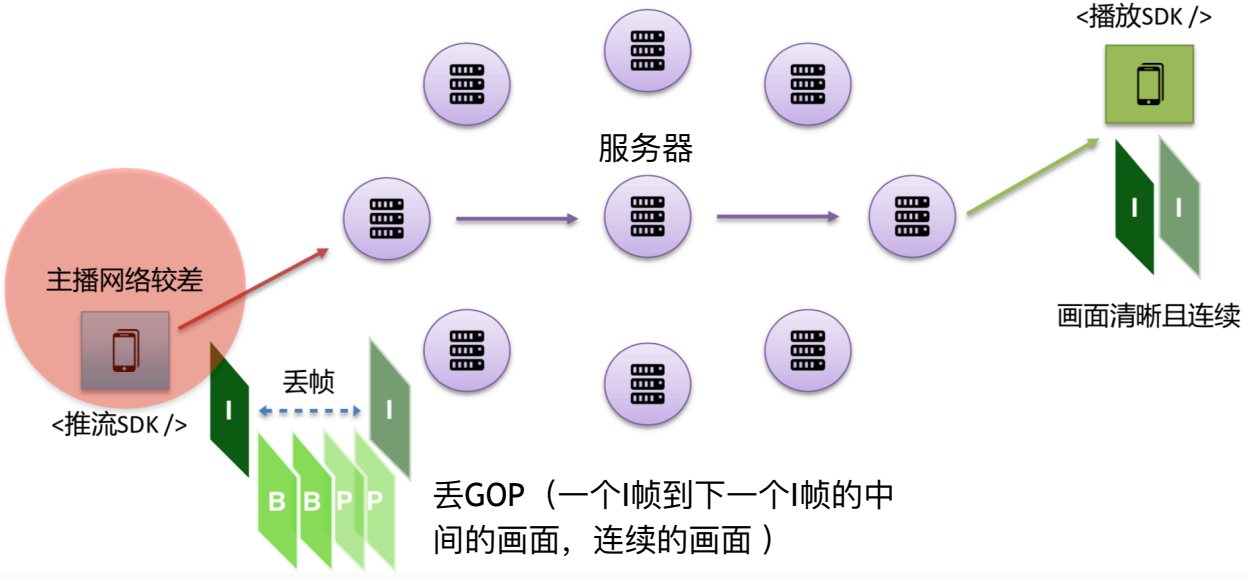
直播技术

- 直播回看安全性
  - 世界杯直播源，被盗链转播，流量失去，广告效果无法保证。
  - 怎么预防
    - 通过加密的Token进行接入访问
    - 基于客户端、网页端反盗链算法
    - 基于大数据信息埋点化，AI训练模型的策略

直播技术

- 网络问题

当主播网络质量较差时，进行弱网丢帧

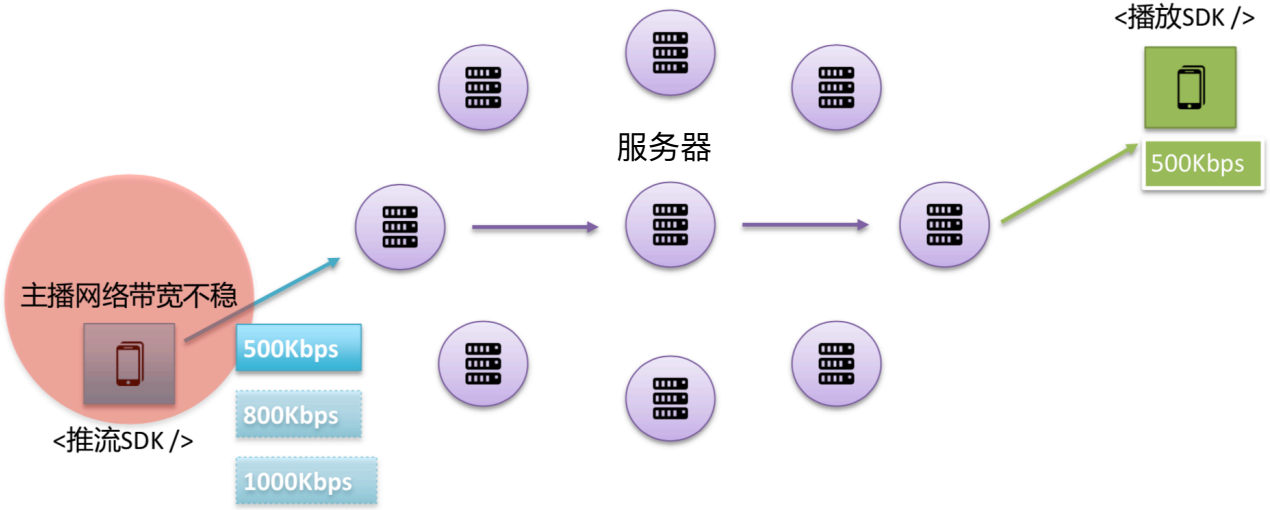


丢GOP（一个I帧到下一个I帧的中间的画面，连续的画面）

直播技术

- 网络问题

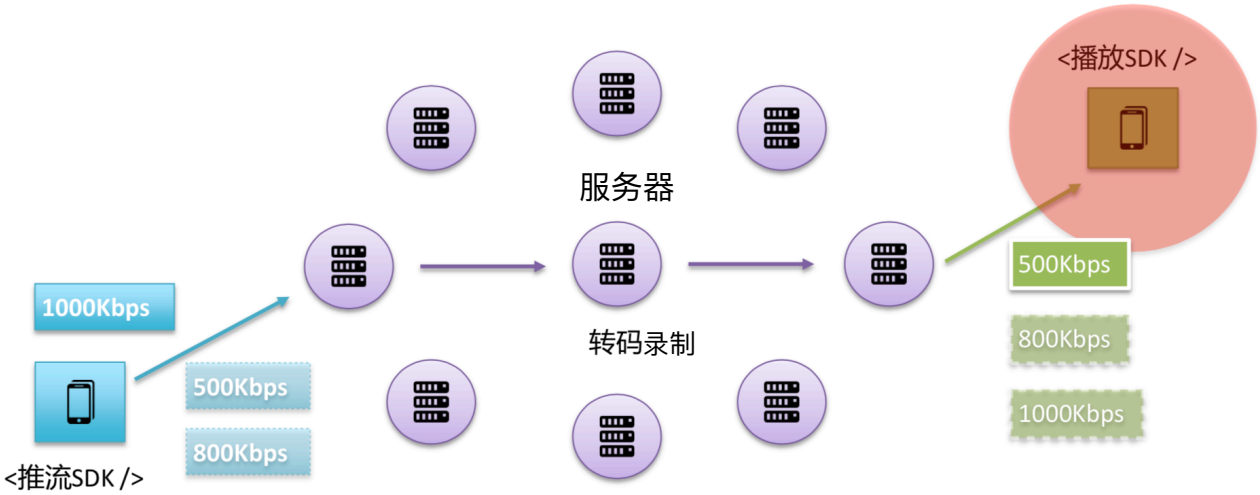
当主播网络带宽不稳时，动态调节推流码率



直播技术

- 网络问题

当观众网络带宽不稳时，播放端动态切换码率



Agenda

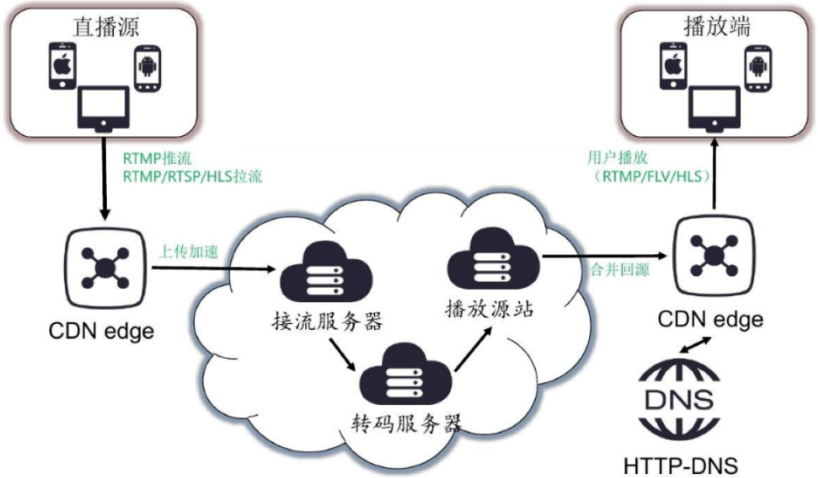
- 流媒体传输协议
- 直播技术
- **音视频通话**
- 音视频引擎构成
- 案例分享
- 常见播放方案

思考一些问题

- 音视频通话与音视频直播有什么不同?
- 为什么音视频通话中有回声？怎么消除回声?
- 音视频通话出问题后，链路怎么分析?

音视频通话

- 音视频通话与音视频直播有什么不同?
  - 协议上：直播一般是RTMP，通话一般是RTP/RTCP
  - 结构上：如图



音视频直播

音视频通话

音视频通话

- 为什么有回声？怎么消除回声？
  - 产生原因：声音一端发送过去后，被采集声音设备采集到，然后回传到另一端
  - 回声消除技术开源的项目有WebRTC和Speex。在这些开源项目之前，回声消除技术是大厂的独门武艺，其它团队只能靠自己一点一滴地摸索积累。在这些开源项目之后，WebRTC和Speex提供开源的AEC模块，成为业界不错的教材。

音视频通话

- 通话链路质量分析
  - 客户端通过埋点进行链路分析，如发起通话，呼叫成功等异常信息分析。

【TraceRoute】(2/7) 101.89.65.217 (101.89.65.217) 56(84) bytes of dat
【TraceRoute】(1/7)当前设备 MODEL:BLA-AL00 BRAND：HUAWEI, TraceRoute: PING
【Android-会议系统模块】会议接通中，当前状态：正在等待对方接受通话
【Android-全链路日志】呼叫成功后，发起后再次加入，LIANGYANSUI311进入通话
【Android-全链路日志】呼叫会议成功，耗时：5510ms
【Android-小鱼SDK模块】小鱼SDK登录成功
【Android-小鱼SDK模块】小鱼SDK开始MakeCall
【Android-全链路日志】收到IM消息：消息类型：CREATE_VIDEOMEETING，是否离线：false,
【Android-小鱼SDK模块】小鱼SDK开始登录
【Android-全链路日志】初始化小鱼SDK，耗时：6ms
【Android-会议系统模块】应用内直接发起音视频点击确认按钮
【全链路日志】聊天页面发起会议
【Android-小鱼SDK模块】小鱼SDK登出
【Android-全链路日志】收到IM消息：消息类型：END_VIDEOMEETING，是否离线：false，会议
【Android-全链路日志】收到IM消息：消息类型：HANGUP，是否离线：false，会议成员：null,
【Android-全链路日志】收到IM消息：消息类型：END_VIDEOMEETING，是否离线：false，会议
【Android-全链路日志】收到IM消息：消息类型：HANGUP，是否离线：false，会议成员：null,
【Android-全链路日志】收到IM消息：消息类型：END_VIDEOMEETING，是否离线：false，会议
【Android-会议系统模块】音视频通话过程正常退出
【Android-小鱼SDK模块】小鱼SDK登出
【Android-小鱼SDK模块】小鱼SDK登出

Agenda

- 流媒体传输协议
- 直播技术
- 音视频通话
- **音视频引擎构成**
- 案例分享
- 常见播放方案

思考一些问题

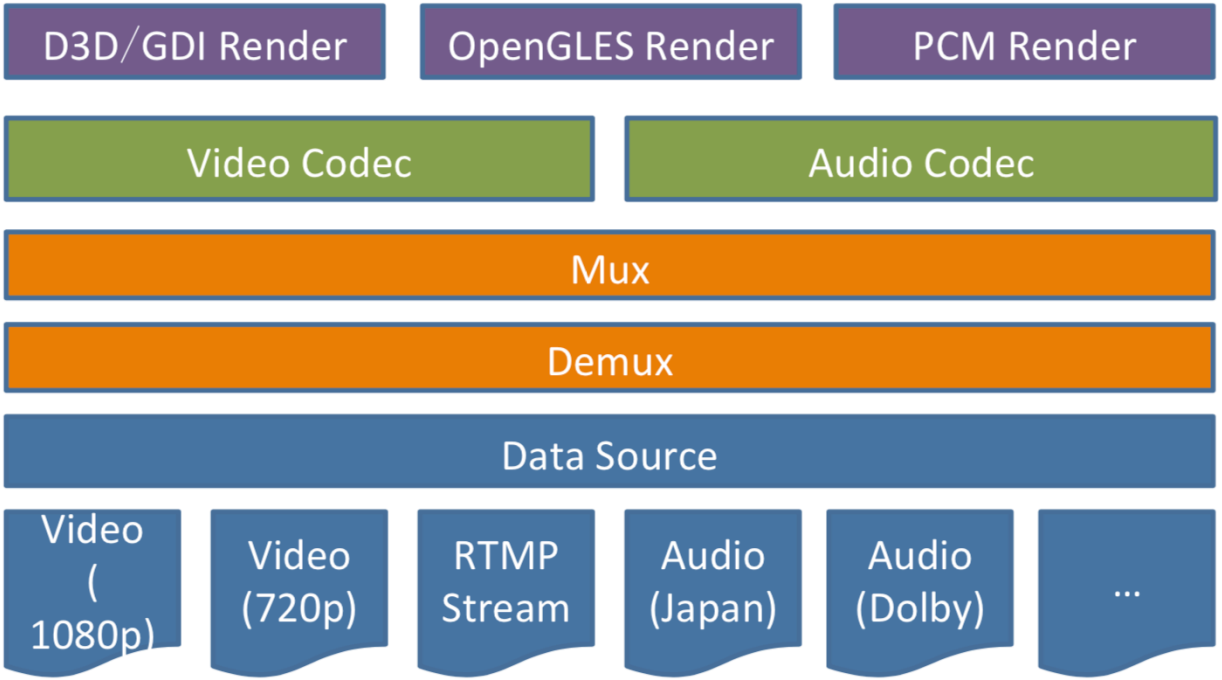
- 你直播，我观看，中间经历了哪些过程?
- 斗鱼、映客、花椒切房间时，怎么能做到那么快直播?
- 网络抖动时，又想看直播，怎么办?
- 工作太忙，但又不想错过xx演唱会直播，怎么回看?
- 网络质量差，怎么让主播和观众稳定使用?

音视频引擎构成

- 整体结构
- 数据请求模块
- 解复用模块
- 解码模块
- 渲染输出模块

音视频引擎构成

- 整体结构
  - 不同引擎部分模块有所区别



音视频引擎构成

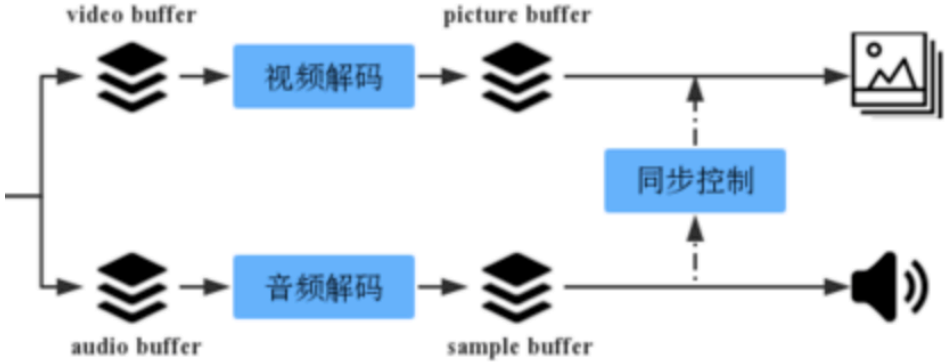
- 数据请求模块
  - 通常就是通过流媒体协议请求数据，流媒体协议包括HLS，RTMP，RTP，RTCP等
  - 请求数据的目的是让播放器可以有足够数据起播，并能持续播放。
  - 数据请求上，可以做本地Server代理操作，作用：一边播放在线视频，一边缓存更多的视频数据，可以提高体验。本地Server，相当于启动一个服务去获取原始视频流地址，然后下载到sdcard上，对播放器提供127.0.0.1这种url地址

音视频引擎构成

- 解复用模块
  - 主要是请求到的数据进行分离出音频流、视频流、字幕流。解复用后，音频流数据会存到音频缓冲队列中，视频流会存到视频缓冲队列中。也就是Packet数据，这时数据还是原始音视频压缩数据。像Android系统中，可以通过MediaExtractor分离出音视频数据。FFmpeg中，也有专门的Demux模块来做解复用。

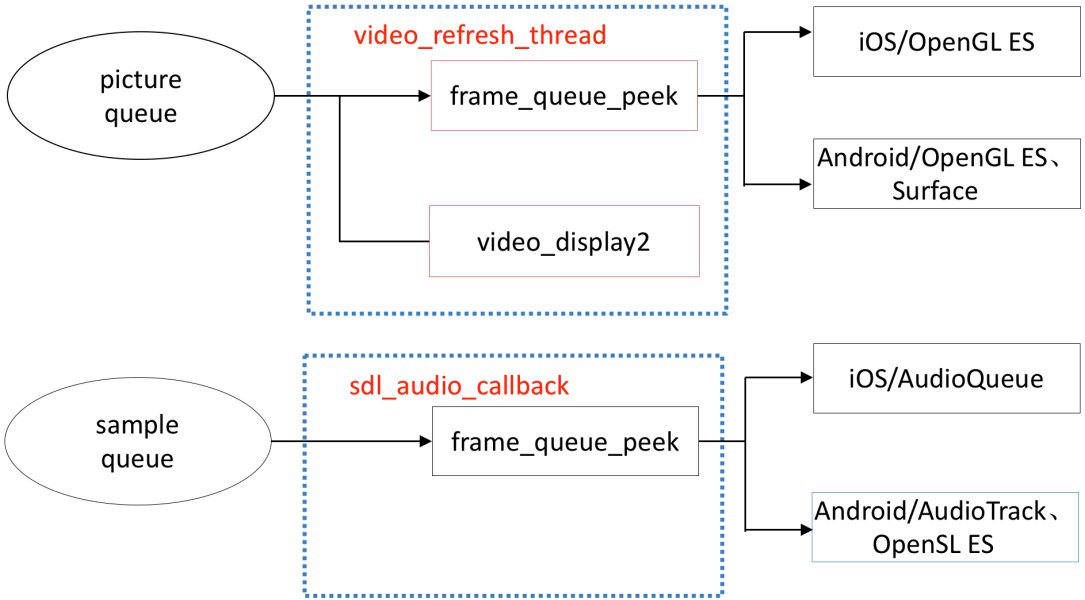
音视频引擎构成

- 解码模块
  - 在解复用分离出音视频后，通知解码器开始去进行解码。解码后通常是一帧一帧数据。最终解码后，视频会到图像队列中，音视频回到采样队列中。然后做音视频同步



音视频引擎构成

- 渲染输出模块
  - 渲染输出是引擎中最后一个模块，主要是将解码后的视频、音频Buffer数据，根据不同平台自有的渲染模块或是，跨平台库进行渲染输出。



Agenda

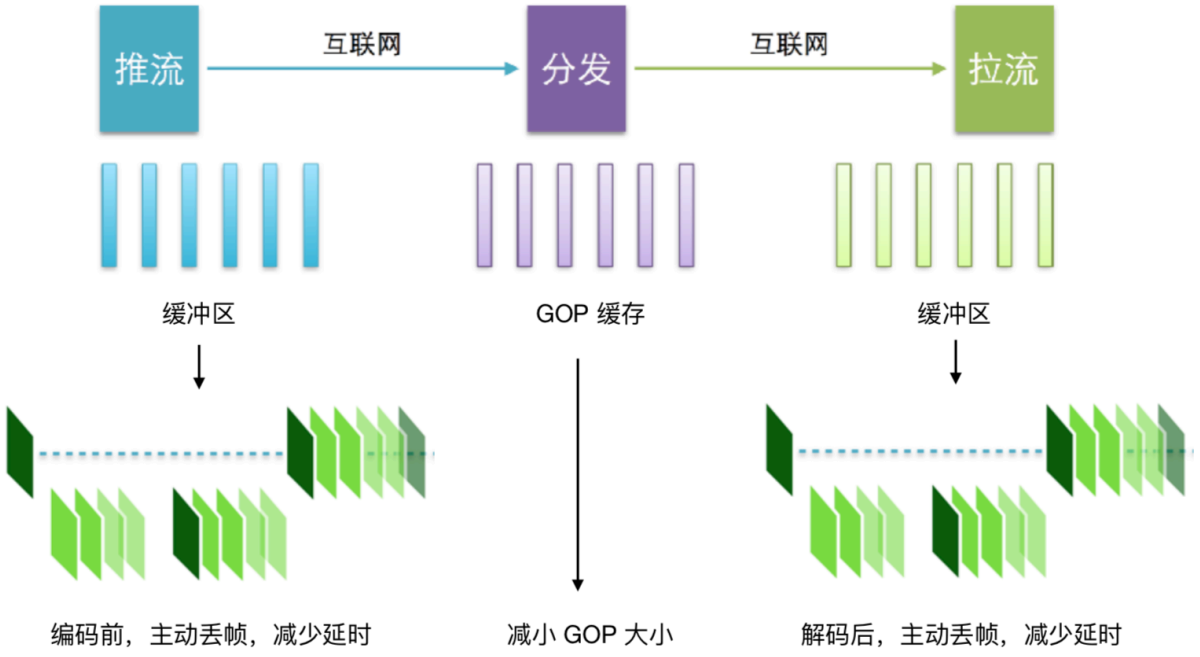
- 流媒体传输协议
- 直播技术
- 音视频通话
- 音视频引擎构成
- **案例分享**
- 常见播放方案

案例分享

- 延时优化
- 音视频不同步
- DNS优化

案例分享

- 延时优化
  - 延时主要来自各业务代码中的缓冲区
  - 整个链路哪些地方导致延时，链路传输过程也是有延时
  - GOP（一个I帧到下一个I帧的中间的画面，连续的画面 ）
  -



案例分享

- 音视频不同步
  - 源的部分视频帧未设置时间戳
    - 在输入输出帧率相同的情况下，会遇到log中大量打印“Past duration XXX too large”且伴有frame drop相关信息的打印。
    - 分析方法：将输入视频包的时间戳信息打印出来，发现一些视频帧未设置DTS/PTS，这些视频帧携带的时间戳都是默认值为AV_NOPTS_VALUE，代码中对该值的定义如下：
    - #define AV_NOPTS_VALUE ((int64_t)UINT64_C(0x8000000000000000))
    - 而这些帧在编码输出之前会被Drop掉，为了满足设定的帧率，后面的有效时间戳的帧会提前输出，导致视频提前而音频滞后。
    - 解决方法：由于我们为这些未设置时间戳的输入帧填充有效时间戳不是一件很容易的事(需要结合输入视频是否采用B帧等)，因此在输入输出帧率相同时，我们可以将这些携带无效时间戳的帧正常送至编码器编码并输出，而不是丢弃。

案例分享

- DNS优化
  - 传统DNS基于UDP，解析时间过长。使用HTTPDNS：预解析、防止域名劫持、精准调度（就近接入）、避免延迟。
  - HTTPDNS主要解决的问题
    - Local DNS 劫持：由于 HTTPDNS是通过 IP 直接请求 HTTP 获取服务器 A 记录地址，不存在向本地运营商询问 domain 解析过程，所以从根本避免了劫持问题。
    - 平均访问延迟下降：由于是 IP 直接访问省掉了一次 domain 解析过程，通过智能算法排序后找到最快节点进行访问。
    - 用户连接失败率下降：通过算法降低以往失败率过高的服务器排序，通过时间近期访问过的数据提高服务器排序，通过历史访问成功记录提高服务器排序。
  - https://help.aliyun.com/document_detail/30140.html?spm=a2c4g.11186623.6.576.iCdgXZ

Agenda

- 流媒体传输协议
- 直播技术
- 音视频通话
- 音视频引擎构成
- 案例分享
- **常见播放方案**

思考一些问题

- 播放音视频需求时，播放器选型怎么选?
- 调研新播放方案时，哪个开源方案可以快速接入?
- 从技术门槛上和应用层使用上，方案的选择?

Agenda

- ExoPlayer
- IjkPlayer
- VLC

常见播放方案-ExoPlayer

- ExoPlayer
  - ExoPlayer 是一个Google开源项目，它不属于Android framework，独立于Android SDK
  - Android支持通过MediaExtractor和MediaCodec实现自定义播放器ExoPlayer介于MediaPlayer和自定义播放器之间的播放器，比MediaPlayer更强的扩展能力
  - ExoPlayer可以通过进一步扩展来处理多种媒体格式
  - 由于它是内置于app中，所以可以随着app进行升级

常见播放方案-ExoPlayer

- ExoPlayer优点
  - 支持 Dynamic Adaptive Streaming over HTTP (DASH) 和SmoothStreaming
  - 支持高级 HLS (HTTP Live Streaming)功能，如正确处理 #EXT-X-DISCONTINUITY的标签

- ExoPlayer缺点
  - ExoPlayer的音频和视频组件依赖Android的 MediaCodec接口，该接口发布于Android4.1（API 16）。因此它不能用于之前的Android版本。

常见播放方案-ExoPlayer

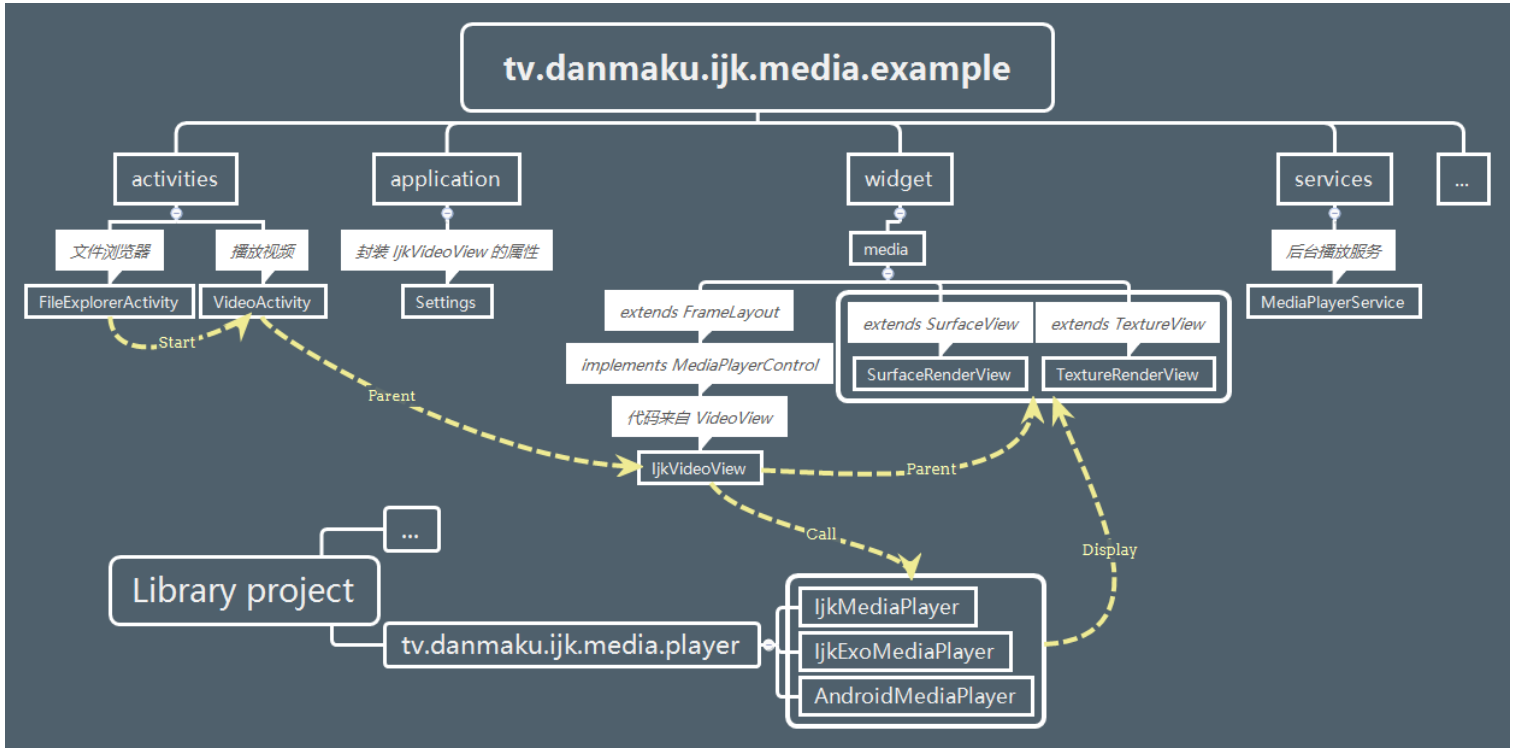
- ExoPlayer
  - YouTube(Android平台客户端)使用ExoPlayer
  - 资料
    - http://developer.android.com/guide/topics/media/exoplayer.html
    - https://github.com/google/ExoPlayer

常见播放方案-Ijkplayer

- Ijkplayer
  - 是基于ffmpeg开源的轻量级视频播放器，支持Android&iOS
  - 封装ffplay、MediaPlayer，MediaCodec，ExoPlayer
  - 支持几乎所有视频封装格式
  - 音频AudioTrack，OpenSL ES

常见播放方案-Ijkplayer

- Ijkplayer



常见播放方案-Ijkplayer

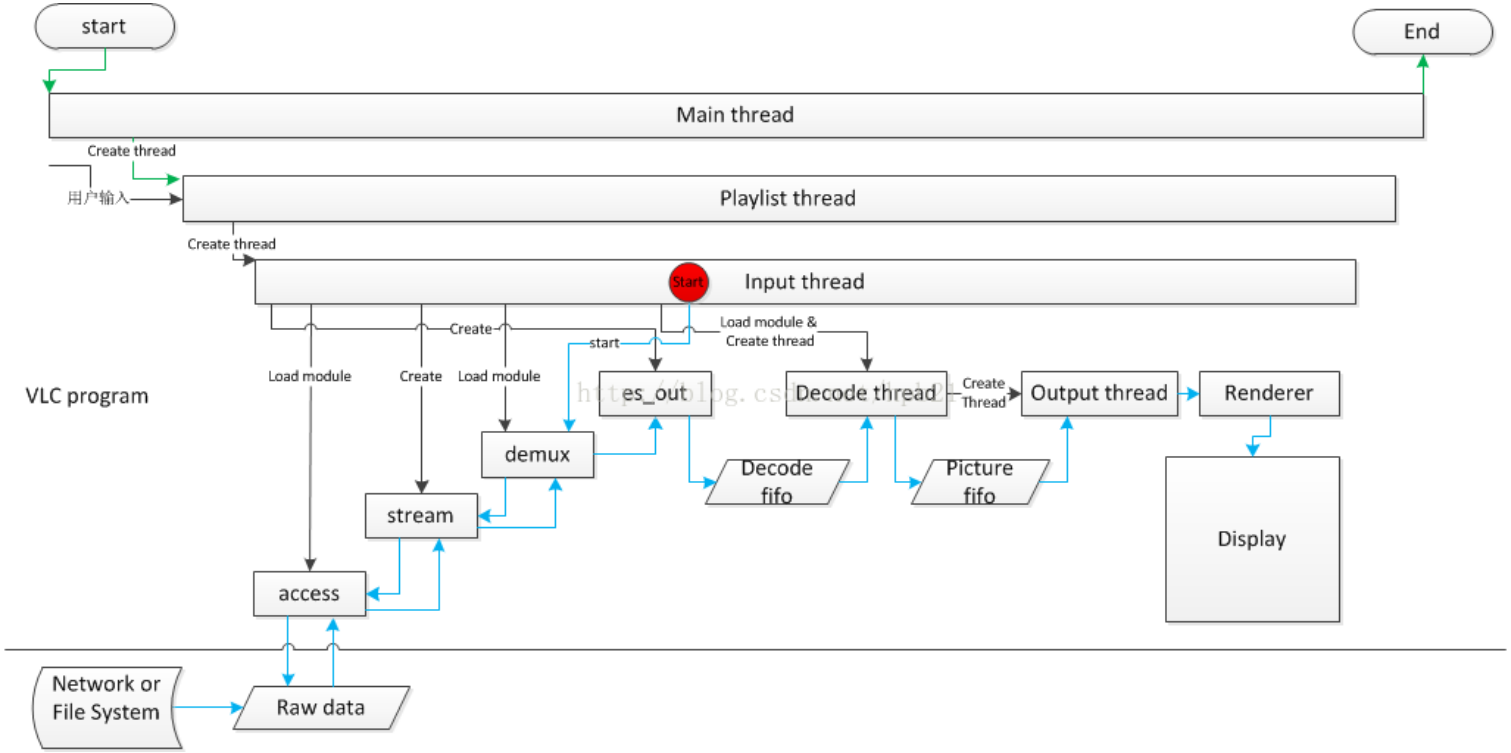
- Ijkplayer
  - B站和主流直播软件都在使用
  - 资料
    - https://github.com/Bilibili/ijkplayer

常见播放方案-VLC

- VLC
  - VLC(Video Lan Client)是一个完整的多媒体框架，最大特点是可以根据需要动态加载许多插件模块，支持视频传输，封装和编码格式。框架核心是利用程序将各模块链接起来。对输入媒体数据，经过各模块处理后输出。
  - 可在所有平台运行 - Windows, Linux, Mac OS X, Unix, iOS, Android ..
  - 支持Dolby，及多音轨，多字幕
  - 内置MediaCodec，ffmpeg，有自己专门的Demux，和自己接入OpenMax组件

常见播放方案-VLC

- VLC



常见播放方案-VLC

- VLC
  - 小米Android ROM集成VLC作为播放器
  - 资料
    - http://www.videolan.org/vlc/
    - git://git.videolan.org/vlc.git

Q&A

谢谢聆听！