# 5G+移动 VR 视频直播

# 设计方案

# 目录

# 1 方案概述

近年来，随着计算机科学和互联网络的普及，使得网络在线影音平台兴起，通过网络进行才艺、游戏、电商等直播成为一种流行的方式。网络直播相较于电视直播等方式有着更便捷的双向互动尤其受到新一代人群的青睐，但人类对美好事物的追求从来都是永无止境的，临场感的缺乏、交互的单一性是传统平面 2D 直播最大的不足，而虚拟现实（VR: Virtual Reality）直播技术的产生解决了这一问题，为用户带来前所未有的沉浸感，促使用户直播体验再上新台阶。

VR 直播，是虚拟现实与直播技术的结合。虚拟现实简称为 VR，指利用电脑模拟三维空间，提供使用关于视觉、听觉、触觉等感官的模拟，让体验者身临其境地观察三维空间。VR 直播与传统平面 2D 直播的差异点在于:

1、提供 180° 或 360° 全景视角，沉浸感更强;

2、通过近眼 3D 显示，使得画面更立体，更真实;

3、可实现更多交互。VR 直播对设备的要求较高，普通的摄像头难以满足要求，需要使用全景拍摄设备捕捉多角度的画面。

用户通过 VR 直播观看可以自由选择任意角度，跳出了传统平面视频的视角框定，由体验者主动选择想观看的内容，而不是被动接受内容，具有体验逼真的沉浸感。

传统体育赛事直播可以给观众带来实时观赛的享受，但只能从特

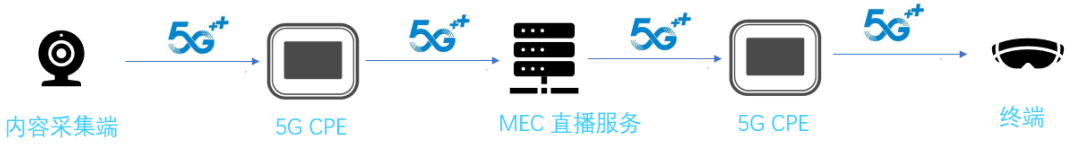
定的角度收看，无法全方位了解现场环境情况。加入 VR 技术后，从视角上拉近了观众与比赛环境的距离，更加为观众营造了现场感的观赛氛围，获得身临其境的观赛体验。体育赛事 VR 直播解决了因现场场地不足、人流量过大导致的场地协调、人员安保、交通疏导等一系列问题，让用户省去了路费、住宿费、现场门票费等巨额费用及大量时间，为用户低成本提供最大限度的赛事直播临场体验。

从经济效益上来说，体育赛事 VR 直播突破了现场座位的限制，大大拓宽了收视群体，还开辟了新的观赛产业——以设备提供、VR 转播权益、VR 广告植入的闭环生态链。

VR 直播行业巨头 NextVR 早在 2015 年就对 NBA 赛事进行了全球首次 VR 直播。在这之后，越来越多的重大体育赛事都采用了 VR 直播技术，包括 2016 年里约奥运会，2018 年俄罗斯世界杯，2018 年平昌冬奥会等。VR 直播这种全新的观影方式正在迅速崛起，为体育赛事注入了新的活力。

# 2 方案设计

## 2.1 整体方案




VR 直播整体方案架构主要由内容采集端、MEC VR 直播服务、网络和终端四个部分构成。

### 2.1.1 直播采集端

VR 直播要求实时的内容生产，内容生产主要包括拍摄、拼接、编码、推流等环节。当前内容生产主要在直播采集端本地完成，采集端部署在直播现场，通常需要配置 VR 摄像机和本地服务器 （用于拼接、编码、推流、导播等），其会对每个机位拍摄的内容生成完整的视频流，再通过导播系统选择某个机位的视频流，经由网络注入 VR 直播平台。视频通过 5G 信号传送至具备算力的 5G CPE，CPE 拼接完成后推流至 MEC VR 直播服务。

### 2.1.2 MEC VR 直播服务

MEC VR 直播服务整体上与传统直播平台相似，核心功能包括：

业务播控/管理：主要负责用户、订购和计费等业务信息的管理，完成用户的接入认证、业务鉴权等业务权限的管控，以及实现节目编排、列表管理等媒资内容的运营。

内容分发：作为 CDN 的源站，实现内容的存储/缓存和分发等。对于云端拼接的场景，平台还需具备拼接处理能力。

CDN 节点：直接面向用户，主要负责内容的缓存与加速、用户视频数据的接入，以及响应调度请求，快速向用户提供媒体流服务。

### 2.1.3 网络

5G 移动通信网络，5G 网络的主要优势在于，数据传输速率远远高于以前的蜂窝网络，最高可达 10Gbit/s，比当前的有线互联网要

快，比先前的 4G LTE 蜂窝网络快 100 倍。另一个优点是较低的网络延迟（更快的响应时间），低于 1 毫秒，而 4G 为 30-70 毫秒。由于数据传输更快，5G 网络将不仅仅为手机提供服务，而且还将成为一般性的家庭和办公网络提供商，与有线网络提供商竞争。以前的蜂窝网络提供了适用于手机的低数据率互联网接入，但是一个手机发射塔不能经济地提供足够的带宽作为家用计算机的一般互联网供应商。

### 2.1.4 终端

在终端侧，用户使用 5G CPE、VR 头盔，通过网络接入 MEC VR 直播服务获取内容，涉及行业场所的多用户接入（2B）、家庭用户的接入（2H）和消费者用户的接入（2C）等多种场景。

## 2.2 硬件方案

### 2.2.1 一体式 VR 摄像机

一体式 VR 摄像机将多个摄像头集成到一个完整的一体化设备中。此类 VR 摄像机方便携带、容易使用，由 VR 摄像机 “一站式”完成处理。

### 2.2.2 5G 智慧 CPE

CPE 作为终端产品之一， CPE 通常被用户称为 5G 移动路由器，在有 5G/4G 信号的地方，可以通过一张 5G/4G SIM 卡提供 Wi-Fi，高

速上网服务，实现无线上网带来的超宽带体验。而 5G 智慧 CPE 则是在原有 CPE 的功能基础上增加算力，通过算力计算完成拼接、推流功能。

### 2.2.3 MEC 边缘计算

本项目硬件集成的边缘计算平台 Edge VLAVR，通过与自研发的视天云的深度协同，将云服务能力延展至边缘侧，抵近真实需求场景，提供安全、可靠、简便易用、功能丰富的边缘计算平台，快速构建并运行各类型边缘业务。

Edge VLAVR 支持多类型应用，并通过编排实现边缘业务的构建:

- 支持多类型边缘应用:包括传统二进制原生类应用、Docker 容器类应用、视频 AI 算法等;
- 支持多模式应用编排编：提供可视化编排工具，用户根据实际业务场景，可自由组合，混合物编排边缘应用，构建边缘业务，并一件下发边部署;
- 支持流式数据处理及数据流编排，帮助用户直接构建基于数据流的边缘业务;

云边协同，一致体验，全流程无缝便捷管理:

- 内置边缘数据模型，与物联网平台数据模型保持一致，终端设备管理无缝衔接;
- 提供多层次监控、远程运维能力:包括边缘软件 OTA 升级、边缘配置升级、边缘应用更新，边缘应用监测;

- 边缘配置支持一键部署到边缘计算节点，部署时能对配置版本进行自动匹配检测；

软件定义，硬件解耦，健壮的本地化支撑能力：

- Edge VLAVR 以软件形式下发部署于硬件节点，与硬件完全解耦，支持轻量化部署；
- 支持离线处理，数据本地处理，提供边端消息离线缓存，支持断网续存；

提供边缘驱动，支持多类型协议硬件设备的边端接入：MQTT、OPC UA、Modbus、ONVIF；

## 2.3 VR 直播服务

VR 直播平台服务基本架构遵循云计算的三种服务模式：

**IaaS 层**：主要包含服务器集群、云存储、云数据库、CDN 与数据管理的能力，实现针对 VR 直播服务进行存储、网络、服务器及相应处理能力的自动部署，具备高度可扩展性。

**PaaS 层**：主要包含网络、播控及大数据等组件，网络组件包括多 CDN 支持、DNS 智能调度等功能；播控组件包括鉴黄、鉴暴、比对等功能；大数据组件包括播放数据反馈分析、CDN 分析等功能。

**SaaS 层**：主要包含内容管理、PGC 直播管理、UGC 直播管理、内容加工等组件，其中：内容管理包括上传管理、编目管理等功能；PGC 直播管理包括频道管理、编码管理等功能；UGC 直播管理包括实时监控、内容检测等功能；内容加工组件包括逻辑打点、在

线快编、云剪辑等功能。

### 2.3.1 直播服务网络要求

相比 VR 点播、VR 游戏等其他的 VR 业务，VR 直播的特殊之处在于内容源从采集端实时上传至云端分发；同时，与 VR 直播观看端仅影响某个用户不同，VR 直播采集端的内容传输影响所有 VR 直播用户的观看，因此需要对 VR 内容的上传带宽要有确定性的保障。

根据 ITU 和 IEEE 的标准，固定宽带非对称 10G PON 可实现上行 2.5Gbit/s、下行 10Gbit/s 的传输速率；而在对称模式中，上下行传输速率均高达 10Gbit/s。

根据 IMT2020(5G)推进组发布的《5G 承载白皮书》，5G 单基站在低频段下行带宽均值为 2.03Gbit/s、峰值为 4.65Gps，上行带宽均值为 691Mbit/s、峰值为 1.55Gbit/s（根据 TDD 上下行占比计算所得），高频频段的带宽速率将更高。

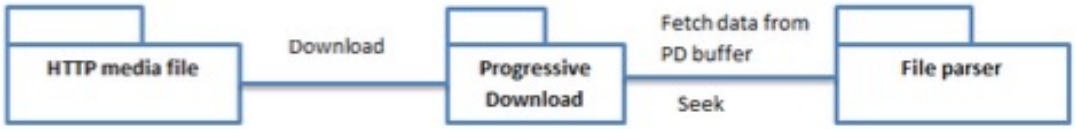
在合理的 10G PON 分光比或 5G 基站接入用户数条件下，固定宽带和移动网络可为 VR 直播用户提供充裕的带宽，确保 VR 直播业务的流畅传输。

## 2.4 视频播放模块

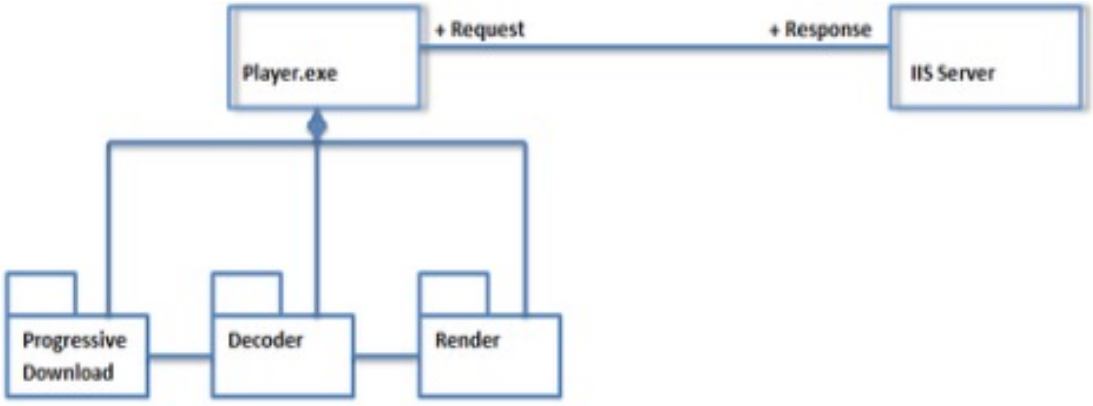
### 2.4.1 流媒体协议

#### 2.4.1.1 渐进式下载协议 Progress Download

渐进式下载是从服务器到客户端的数字媒体文件传输，通常是在从计算机启动时使用 HTTP 协议进行传输终。终端用户可以在下载完成之前开始播放媒体。改播放器将媒体数据从 http 服务器下载到本地缓冲区（例如本地内存或文件），有效地管理有限的缓冲区，然后根据需要向文件解析器提供所需的数据。




要运行渐进式下载，用户需要将至少 40 MB 的正在播放的媒体文件下载到设备的内存中。如果所需的媒体数据不在缓冲区中，则播放器从 HTTP 服务器下载数据。如果所需的媒体数据在缓冲区中，则播放器从本地缓冲区读取并返回数据。



#### 2.4.1.2 实时流协议 RTSP

实时流协议（RTSP）是一种网络控制协议，旨在用于娱乐和通信系统中以控制流媒体服务器。该协议用于在端点之间建立和控制媒体会话。媒体服务器的客户端发出 VCR 样式的命令（例如播放和暂停），以便于实时控制服务器中媒体文件的播放。流数据本身的传输不是 RTSP 协议的任务。大多数 RTSP 服务器使用实时传输协议（RTP）和实时控制协议（RTCP）进行媒体流传递。但是，某些供应商实现专有的传输协议。 例如，RealNetworks 的 RTSP 服务器软件还使用了 RealNetworks 专有的 Real Data Transport（RDT）。

播放器支持 RTSP 中如下的音频和视频解码：

1. Audio
    - MPEG4-GENERIC
    - MPEG4-LATM
    - AMR-WB+
    - AMR-WB
    - AMR-NB
    - QCELP
    - WMA
    - Video
    - H.264
    - MPEG4

- H.263
- WMV

播放器中的 RTSP 支持下列相关协议：

1. RFC 2326: Real Time Streaming Protocol (RTSP)
2. RFC 2327: SDP: Session Description Protocol
3. RFC 3550: RTP: A Transport Protocol for Real-Time Applications

下面是支持的 RTP 协属性：

1. RFC 3551: RTP Profile for Audio and Video Conferences with Minimal Control

以下是支持的 RTP 有效负载类型：

1. RFC 3984: RTP Payload Format for H.264 Video
2. RFC 3016: RTP Payload Format for MPEG-4 Audio/Visual Streams
3. RFC 4629: RTP Payload Format for ITU-T Rec.H.263 Video
4. RFC 3267: Real-Time Transport Protocol (RTP) Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (ARM-WB) Audio Codecs
5. RFC 4352: RTP Payload Format for the Extended Adaptive Multi-Rate Wideband (AMR-WB+) Audio Codec
6. RFC 2658: RTP Payload Format for PureVoice (tm) Audio
7. MS-RTSP: RTP Payload Format for ASF Streams

#### 2.4.1.3 Windows Media HTTP

Windows Media HTTP（WMHTTP）流协议是基于客户端/服务器的协议，用于在客户端（流数据的接收方）和服务器（流数据的发送方）之间流实时数据。播放器支持以下 WMHTTP 功能：

1. 支持非管道模式和管道模式
   - 非管道模式：一种操作模式，其中来自客户端的请求必须通过与服务器用于将内容流式传输到客户端的 TCP 连接分开的 TCP 连接发送；
   - 管道模式：一种操作模式，在该模式下，可以在服务器用于将内容流式传输到客户端的同一 TCP 连接上发送来自客户端的请求。
2. 支持单文件流和实时流
   - 单文件流传输：有关更多信息，请参见单文件流传输；
   - 实时流式传输：在仍由编码器编码时流式传输的内容。
3. 指定传输速率
   - 指定多媒体数据量（以毫秒为单位），客户端请求服务器加快速度；
   - 保持在线请求可防止服务器超时；
   - 将有关流内容的统计信息提交到服务器。

#### 2.4.1.4 MPEG-OMAF

Omnidirectional MediA Format (OMAF) 全向媒体格式是由 Moving Picture Expert Group (MPEG) 运动图像专家组所打造的 360 度媒体内容标准。OMAF 为全向内容具定义了 3 个级别的自由度 (3DOF)，例如 360° 视频，图像，音频和定时文本。并且将与视口无关的 360° 视频转换为符合 OMAF 的内容仅需要文件格式和传输协议级别的修改（例如，基于 MP4 和 DASH 的分段流）。

OMAF HEVC“视口相关”的基础提高了视频编码的要求，因为它用不同的质量和分辨率对前景（视口）和背景进行编码。视口自适应操作通过将重点集中在用户正在观看的区域上，减少 360° 视频所需的带宽。由于用户可以自由转动头部并更改活动视口，因此要求系统必须能够快速做出反应，以把不同区域的视频切换到高质量。

从解码和播放渲染客户端来说，OMAF Player 只需要知道 DASH 清单的 URL 就可以开始播放。从本质而言还是一个 DASH 的播放器，OMAF 将与 VR 相关的元数据添加到 ISOBMFF 和 DASH 清单中，播放器根据当前 IMU 融合的空间姿态四元数与 OMAF 将与 VR 相关的元数据比对获取相应质量的高清播放流。

### 2.4.2 自适应流传输模式

自适应流传输是一种用于通过计算机网络流传输多媒体的技术，该技术通过实时检测用户的带宽和 CPU 容量并相应地调整视频流的

质量来工作。它需要使用能够以多个比特率编码单个源视频的编码器。 播放器会根据可用资源在流式传输不同编码之间进行切换。自适应流包括以下三种类型：

1. HTTP 实时流（HLS）
2. Microsoft 平滑流
3. HTTP 上的动态自适应流（DASH）

#### 2.4.2.1 HLS —— HTTP Live Streaming

HTTP Live Streaming（缩写是 HLS）是由苹果公司提出基于 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。在开始一个流媒体会话时，客户端会下载一个包含元数据的 extended M3U (m3u8) playlist 文件，用于寻找可用的媒体流。

HLS 只请求基本的 HTTP 报文，与实时传输协议（RTP）不同，HLS 可以穿过任何允许 HTTP 数据通过的防火墙或者代理服务器。它也很容易使用内容分发网络来传输媒体流。

相对于常见的流媒体直播协议，例如 RTMP 协议、RTSP 协议、MMS 协议等，HLS 直播最大的不同在于，直播客户端获取到的，并不是一个完整的数据流。HLS 协议在服务器端将直播数据流存储为

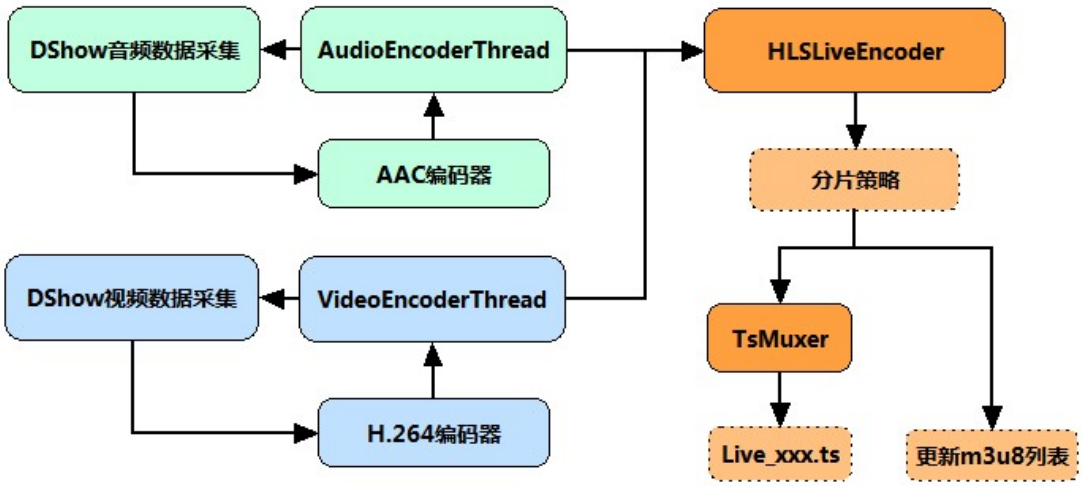
连续的、很短时长的媒体文件（MPEG-TS 格式），而客户端则不断的下载并播放这些小文件，因为服务器端总是会将最新的直播数据生成新的小文件，这样客户端只要不停的按顺序播放从服务器获取到的文件，就实现了直播。由此可见，基本上可以认为，HLS 是以点播的技术方式来实现直播。由于数据通过 HTTP 协议传输，所以完全不用考虑防火墙或者代理的问题，而且分段文件的时长很短，客户端可以很快的选择和切换码率，以适应不同带宽条件下的播放。不过 HLS 的这种技术特点，决定了它的延迟一般总是会高于普通的流媒体直播协议。

根据以上的了解要实现 HTTP Live Streaming 直播，需要研究并实现以下技术关键点：

1. 采集视频源和音频源的数据
2. 对原始数据进行 H264 编码和 AAC 编码
3. 视频和音频数据封装为 MPEG-TS 包
4. HLS 分段生成策略及 m3u8 索引文件
5. HTTP 传输协议

通过以上分析，实现 HLS LiveEncoder 直播编码器，其逻辑和流程基本上很清楚了：分别开启音频与视频编码线程，通过 DirectShow（或其他）技术来实现音视频采集，随后分别调用 libx264 和 libfaac 进行视频和音频编码。两个编码线程实时编码音视频数据后，根据自定义的分片策略，存储在某个 MPEG-TS 格式分段文件中，当完成一个分段文件的存储后，更新 m3u8 索引文件。

如下图所示：




上图中 HLSLiveEncoder 当收到视频和音频数据后，需要首先判断，当前分片是否应该结束，并创建新分片，以延续 TS 分片的不断生成。需要注意的是，新的分片，应当从关键帧开始，防止播放器解码失败。核心代码如下所示：

```
        // 从关键帧开始
        if (isKeyframe)
        {
            // 检查是否应当创建新的分片文件
            if (CheckRecordTimestamp(timestamp))
            {
                if (ts_muxer_)
                {
                    std::string tsfilename = ts_muxer_->Filename();
                    ts_muxer_->WriteEnd();
                    delete ts_muxer_;
                    ts_muxer_ = 0;

                    // 更新m3u8索引文件
                    UpdateM3U8List(tsfilename, real_duration_/1000);
                }

                // 生成新的分片文件名
                std::string newfilename = GetRecordFilename();
                ts_muxer_ = new TSMuxer(newfilename.c_str());
                ts_muxer_->WriterHeader();
            }
        }
```

TsMuxer 的接口也是比较简单的。

```cpp
#include "mpegts/mpegtsenc.h"

#define _USE_TS_MUXER_LOG_

class TSMuxer
{
public:
    explicit TSMuxer(const char* filename);

    ~TSMuxer();

    std::string Filename() { return filename_; }

    // 初始化，创建MPEG-TS音频和视频输出流
    void WriterHeader();

    // 将内存中剩余数据写入文件
    void WriteEnd();

    // 写入视频数据
    void WriteVideoData(char* buf, int bufLen, bool isKeyframe, unsigned int timestamp);

    // 写入音频数据
    void WriteAudioData(char* buf, int bufLen, unsigned int timestamp);

    // 废弃
    void WriteVideoSeqheader(char* buf, int bufLen);

private:
    std::string filename_;
    __int64 time_begin_;
    FILE* fp_;
    bool is_video_begin_;
    MpegTSWrite* mpegts_write_;
```

苹果公司把 HLS 协议作为一个互联网草案（逐步提交），在第一阶段中已作为一个非正式的标准提交到 IETF。2017 年 8 月，RFC 8216 发布，描述了 HLS 协议第 7 版的定义。

以下是播放器支持的标签。

- EXT-X-BYTERANGE
- EXT-X-DATERANGE
- EXT-X-DEFINE
- EXT-X-DISCONTINUITY
- EXT-X-ENDLIST
- EXT-X-I-FRAME-STREAM-INF
- EXT-X-I-FRAMES-ONLY
- EXT-X-INDEPENDENT-SEGMENTS

- EXT-X-KEY(IV)
- EXT-X-KEY(URI)
- EXT-X-MAP
- EXT-X-MEDIA
- EXT-X-MEDIA(SUBTITLED/FORCED/CHARACTERISTICS)
- EXT-X-MEDIA-SEQUENCE
- EXT-X-PLAYLIST-TYPE
- EXT-X-PROGRAM-DATA-TIME
- EXT-X-SESSION-DATA
- EXT-X-SESSION-KEY
- EXT-X-START
- EXT-X-STREAM-INF
- EXT-X-STREAM-INF(AUDIO/VIDEO)
- EXT-X-STREAM-INF(SUBTITLES)
- EXT-X-TARGETDURATION
- EXT-X-VERSION

播放器支持以下在 HLS 里的媒体格式

- MPEG2-TS
- MPEG-ES/AAC
- MPEG-ES/MP3
- ID3(MPEG-ES)
- ID3(MPEG2-TS)

- WebVTT

#### 2.4.2.2 DASH —— HTTP 动态自适应流

HTTP 动态自适应流（DASH），也称为 MPEG-DASH，是一种自适应比特率流技术，可通过 Internet 从常规 HTTP Web 服务器提供高质量的媒体内容流。与 Apple 的 HTTP Live Streaming（HLS）解决方案类似，MPEG-DASH 的工作原理是将内容分成一系列小的基于 HTTP 的文件段，每个段包含一段短时间的内容回放时间，这可能会持续数小时，例如电影或体育赛事的直播。

当内容由 MPEG-DASH 客户端播放时，客户端会根据当前的网络状况自动从替代项中选择下一个要下载和播放的片段。客户端选择可能具有最高比特率的段，可以及时下载该段进行回放，而不会引起停顿或重新缓冲事件。因此，MPEG-DASH 客户端可以无缝地适应不断变化的网络状况，并提供高质量的播放而不会出现停顿或重新缓冲事件的情况。

DASH 是由 MPEG (Moving Picture Experts Group)组织制定，2010 年开始启动，2011 年 11 月发布 Draft 版本，2012 年 4 月发布第一稿 Version（ISO/IEC 23009-1:2012），2014 年 5 月发布第二稿（ISO/IEC 23009-1:2014），最新稿（ISO/IEC 23009-3:2015）。

目前 3GPP Release 10 已经将 DASH 纳入其中；在 HbbTV 1.5 中也支持 DASH；DVB-DASH 也将 DASH 纳入到 DVB（ETSI TS 103 285 v.1.1.1）。目前 DASH Industry Forum 由发起厂家组成，致力于推

进 DASH 产品生态，将 DASH 产业化和业界最佳实践推向批量应用。

ISO/IEC 23009 包含四个部分:

- Part 1: Media presentation description and segment formats
- Part 2: Conformance and reference software
- Part 3: Implementation guidelines
- Part 4: Segment encryption and authentication

DASH 相比 HLS、HSS 等协议的优势在于:

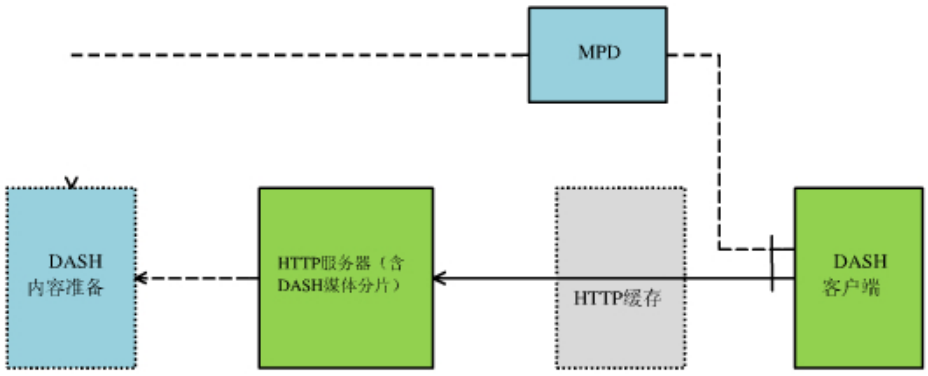
1) DASH 支持多种编码，支持 H.265、H.264、VP9 等等;
2) DASH 支持 MultiDRM，支持 PlayReady、Widewine，采用通用加密技术，支持终端自带 DRM，可以大幅度降低 DRM 投资成本;
3) DASH 支持多种文件封装，支持 MPEG-4、MPEG-2 TS（Transport Stream）;
4) DASH 支持多种 CDN 对接，采用相同的封装描述对接多厂家 CDN;
5) DASH 支持异构终端，浏览器原生不用插件就可以支持，Android/iOS/Windows/Flash 可以通过 JITP 将 DASH 转换为 HLS、HDS、HSS 等，已支持 Legacy 终端类型，支持一份存储，大幅度减少文件存储量;
6) DASH 支持直播、点播、录制、时移等等丰富的视频特性;
7) DASH 支持动态码率适配，支持多码率平滑切换;

8) DASH 支持紧缩型描述以支持快速启动；

9) DASH 支持客户端和服务端的广告插入。

DASH 典型的一个端到端的系统包含 Encoder、Dash Server、Origin Server、Edge Server、DASH Client：

Media Presentation Description 是描述分片和时序的信息：从 MPD->Period->Adpatation Set(Video/Audio Track)->Representation(Multiple bitrate)->Segement




DASH 内容准备提供不同码率的视频文件（使用不同的质量需求和网络环境），然后使用特定的分片方法， 将视频文件分片，然后分片传输到客户端进行播放。MPD 文件时在视频文件进行分片时获得的视频本身的属性信息和视频分片信息。

DASH 支持以下视频和音频编解码器：

1. Video codec

- H.264
- WMV/VC1
- H.265

2. Audio codec

- AAC
- WMA

DASH 支持的功能包括：

- Request and response through HTTP
- XML parser used for Media Presentation Description (MPD) file
- Audio and video fragment process mechanism
- MP4 fragment parser callback
- TS fragment parser callback
- Playback of multi-period VoD content
- Playback of indexed content
- Playback of PD content (one file)
- Playback of live content
- Playback of VoD TTML or SMPTE subtitle

DASH 使用 ISO，IEC 和 23009 标签。媒体表示描述（MPD）文档包含 DASH 客户端所需的元数据，以构造适当的 HTTP URL 来访问段并向用户提供流服务。

Media Presentation 包含一个或多个 Periods。Multi_Period 用于广告插入。

- AdaptationSet：每个 Period 包含一个或多个 AdaptationSet。通常，AdaptionSet 分为三种类型的流：视频，音频和字；
- ContentComponent：每个 AdaptationSet 包含一个或多个媒体

ContentComponents。每个媒体 ContentComponent 的属性由 ContentComponent 元素描述，或者如果 Adaptation Set 中仅存在一个媒体 ContentComponent，则可以直接在 AdaptationSet 元素上描述。

一个 Presentation 是在定义的期间内包含媒体内容的媒体内容组件的完整集合或子集的替代选择之一。

- Segment 由 SegmentTemplate 元素定义。在这种情况下，特定的标识符将由分配给细分的动态值代替，以创建细分列表；
- SegmentTimeline 元素表示表示形式中每个段的最早表示时间和表示持续时间（以基于@timescale 属性为单位）。

#### 2.4.2.3 比特率自适应

比特率自适应是一种在各种环境条件下有效传递流内容以获得最佳播放质量的过程。播放器支持基于 HTTP 的自适应流协议（即 HLS，平滑流和 DASH）的比特率自适应。播放器考虑了网络拥塞，设备功能，设备负载以及其他几个因素，以确定要播放的最佳视频比特率。自适应功能可在不同视频质量之间无缝过渡，而不会中断播放。播放器遵循以下规则来执行比特率适配操作：

- HLS: 播放器选择默认的视频/音频轨道；
- SS/DASH: 播放器选择清单文件中定义的第一个视频/音频轨道。

除了自动比特率自适应以外，OSMP +还支持许多配置选项以完

善自适应行为。这些配置选项包括：

- 初始（开始）比特率
- 开始缓冲时间
- 播放重新缓冲时间
- 最大缓冲时间
- 最小比特率
- 最大比特率
- 基于 CPU 的适应
- 设备功能配置

将这些选项与自动比特率自适应技术相结合，可提供一种灵活的解决方案，以满足各种需求。

### 2.4.3 IPV6-only 网络支持

仅 IPv6 网络支持在 iOS 和 Mac OS 上可用。

除了支持 IPv4 或双栈（IPv4 / IPv6）网络外，Apple 还要求为提交给 App Store 的所有应用程序支持仅 IPv6 网络。

播放器与从此版本开始的 DNS64 / NAT64 转换机制兼容，以便在客户端仅具有 IPv6 连接并尝试寻址 IPv4 上托管的内容时容纳工作流程。

如果用户尝试从仅 IPv6 的网络访问托管在 IPv4 上的内容，它将请求 DNS64 从 URL 域名获取 IP 地址，并在 IPv6 空间中接收表示 IPv4 地址的合成 IPv6 地址。使用此综合地址，用户可以使用

NAT64 建立到 IPv4 主机的连接，该连接将 IPv6 网络连接到 IPv4 网络。

另一方面，如果用户尝试从纯 IPv6 网络访问托管在 IPv6 上的内容，则它使用 DNS64 获取 URL 域的 IPV6 地址，并使用标准 IPv6 路由机制建立与内容的 IPv6 连接。

在下列平台上，此新功能可用于清除和加密内容：

- iOS 9.0 + / OS X 10.11.4+：同时支持 IPv4 内容的 URL 主机地址和 IP 文字地址；
- 低于 iOS 9.0 或 OS X 10.11.4 的版本：仅支持 IPv4 内容的 URL 主机地址。

以下是日志消息示例，可帮助您识别/验证此 DNS64 / NAT64 转换。

- 如果 IPv4 内容是 URL 主机地址，则您可以使用以下格式获取日志：

```
1. '15:08:38.319 @@@VOLOG, Info, ModuleID [xxxxxxxx], ThreadID
   [xxxxxxxx], vo_http_stream.cpp, send_request, Line#1731,
   [SourceIO] address:64:ff9b::a02:4407, url:example.visualon.com'
```

- 如果 IPv4 内容是 IP 地址，则您可以使用以下格式获取日志：

```
1. 'vo_socket.cpp, VO_IOS_DNSResolve, Line#626, IP item: i=0,
   host=10.2.68.7, port=8082, is_ipv6=1, addr=64:ff9b::a02:4407,
   dataLen=28, sa_len=28, fam=30'.
```

### 2.4.4 低延时和同步播放

播放器在 Android，iOS 和 HTML5 +平台上支持 DASH 实时流低延迟特性。有两种类型的 API，第一种类型用于启用低延迟模式；第二种类型用于设置目标等待时间值（同步回放）。请参考下表的低延迟 API 和表现。

| API 类型 | APIs 和 sourceConfig | 行为 |
|---|---|---|
| 1. 打开低延时 API | enableLowLatencyMode | 当 API 打开时，低延时模式打开，默认值为 3000 毫秒 |
| 2. 设置低延时的值 | SetPresentationDelay | When the API is set, synchronization playback mode is on, and the target latency is the value set by this API 当 API 打开时，同步模式打开，然后用户可以修改此值。 |

API 的第二种类型是设置目标延迟值。(例如播放器上的 setPresentationDelay（） API)。该值指示在实际直播点之前设置了多少时间。通常，将目标等待时间设置为较低的值可以减少直播过程中的实际等待时间，但是，如果网络带宽不足，则将目标等待时

间设置为低于 3000 毫秒可能会导致频繁加载的性能问题。 如果目标等待时间值设置为 0，则播放器将非常积极的赶上实时点。

延迟的目标持续时间是由 setPresentationDelay（）设置的值决定，该值指示在实际现场播放点之前设置了多少时间。默认情况下，setPresentationDelay（）的值为 3000 毫秒。

为了帮助在不同设备上同步播放 DASH 内容，可以使用以下机制来定义公共时间参考。如果 DASH 清单中有 UTCTiming 标签，它将用作同步的时间参考。 默认情况下启用。如果清单中有多个 UTCTiming 标签，则将考虑第一个 UTCTiming 标签，其他标签将被忽略。如果 DASH 清单中没有 UTCTiming 标签，则除非应用程序设置了另一个网络时间协议（NTP）时间参考，否则播放器将使用 http://time.akamai.com/?iso&ms 中的 UTC 时间。

要设置 UTC 时间，应用程序必须使用 addConfiguration 提供 NTP 时间方案和值。

例如，以下键值对可以作为 addConfiguration 的 JSON 字符串参数（确保将转义字符配置为适合目标 OS）：

```
1. {
2. "schemeIdUri": "urn:mpeg:dash:utc:http-head:2014", "value":
   "https://time.is/"
3. }
4. or
5. {
```

```
6. "schemeIdUri": "urn:mpeg:dash:utc:http-xsdate:2014",
7. "value": "https://time.is/"
8. }
```

以下是播放器中时间信息的说明示例。下图中的时间信息已在黄色进度条上从 T1 到 T6 标记，与上面的时间相对应。


- T1：启用“实时流 DVR 位置”时，该值始终为 DVR 窗口持续时间，表示最小实时位置；当禁用“实时流 DVR 位置”时，该值从 0 开始，并随着播放 的进行而平稳地增加；
- T2：该值是 UTC 时间与 UTC 播放时间之间的时间差。禁用“实时流 DVR 位置”时，该值从 0 开始并平稳增加；
- T3：UTC 播放位置；

- T4：UTC 时间；
- T5：PTS 位置（以秒为单位）；
- T6：启用“实时流 DVR 位置”时，该值始终为 0，表示最大实时位置，也表示实时位置；禁用“实时流 DVR 位置”时，该值从 DVR 窗口时间开始并持续增加。

# 3 产品兼容性及关键技术指标

## 3.1 产品功能的复杂性和优势

本方案所基本技术指标如下：

| 模块 | 功能 | 要求 |
| --- | --- | --- |
| VR直播 | 接收直播信号 | 接收RTMP推/拉流 |
| | | 接收HLS拉流 |
| | 接收编码类型 | 接收H.264、H.265编码格式视频流 |
| | 自定义编码库 | 自定义直播编码参数，用户可自定义视频分辨率、视频帧率、视频码率、音频码率 |
| | 直播流透传 | 支持VR直播流透传，可将RTMP透传为RTMP、HLS，HLS透传为RTMP、HLS |
| | 多协议VR直播流输出 | 一路输入VR流可同时转出RTMP、HLS多种类型VR视频流 |
| | 多码率输出 | HLS 直播流输出时支持多码率输出，可同时输出多种不同码率HLS视频流 |
| | 分辨率支持范围 | 平台支持4K~CIF分辨率VR视频流 |

系统具体有如下特点：

1. **内容回传（上行网络）**：使用 5G 或专线，VR 直播要求采集端内容实时、稳定地回传至云端，需要高品质的上行网络。如果 VR 直播采集现场支持固定宽带接入（有线或 Wi-Fi），建议采用高性能 ONT 接入+专线传输，为 VR 直播回传提供专用带

宽。如果 VR 直播在户外或其他难以接入高质量宽带的场所，建议采用 5G 接入、传输。

2. **内容分发（下行网络）：**2B 场景优选专线，2H 场景优选千兆家宽 VR 直播内容分发包括 CDN 中心节点向边缘节点分发和 CDN 节点向终端分发。其中，CDN 中心节点向边缘节点的内容分发采用专线传输，按需扩大带宽；而 CDN 节点向终端的内容分发按接入的场景类型，采用不同的网络方案。

(1) 2B 场景

对于 2B 场景，如 VR 直播“第二现场”、分会场、VIP 包厢等场所，有多用户同时接入，并发码率大，建议 CDN 节点与场所间配置专线保障。当 VR 终端数量规模不太大（如 20 人以内）时，优先选择 Wi-Fi 接入，需要部署多个高性能 AP 并对 Wi-Fi 频段进行合理规划，避免干扰。当 VR 终端数量规模太大时，Wi-Fi 频段资源有限，过大的视频流量并发，极易引发 Wi-Fi 干扰，影响用户体验，此时建议 VR 终端采用 USB 转网口线缆连接网络设备，采用有线推流的方式，以保障多人同时在线的体验。

随着 Wi-Fi 6（802.11ax）的应用，其通过 MU-MIMO、OFDMA 频分复用、1024-QAM 高阶编码等技术，从频谱资源利用、抗干扰等方面提升接入容量和传输效率，结合业界领先的家庭网关和专业的家庭组网能力，可实现 Wi-Fi 实现更多用户接入并保障 VR 直播体验。

(2) 2H 场景

对于 2H（家庭）场景，通过配以千兆家宽并采用高性能 5GHz Wi-Fi 接入，保障 VR 直播观看体验。

(3) 2C 场景

C 端用户可通过集成 5G 模块的终端在室外接入观看 VR 直播。当前已有手机支持 5G，但 VR 头盔尚未支持。

## 3.2 性能指标和优化

相比传统直播，VR 直播内容码率更大，需要高品质网络来确保内容回传和分发过程中视频的高效传输，保障业务体验。当前 VR 直播主要内容规格的建议码率和带宽如下图所示。

| VR直播场景 | 建议码率 | 建议带宽 |
| --- | --- | --- |
| 8K VR（2D/3D） | 80~120 Mbit/s（采用全视角传输） | 120~180 Mbit/s |
| 4K VR（2D/3D） | 20~40 Mbit/s | 30~60 Mbit/s |

8K VR 采用全视角传输时，对 VR 终端解码能力要求较高；当前具备 8K 解码能力的终端较少，受限于终端的 解码能力，可考虑采用 FoV 传输方案。以 8K 2D VR 为例，若采用 TWS 方案进行 FoV 传输，低清背景流码率约 6~15Mbps，高清 Tile 流总和约 80Mbps，网络传输的是背景流和 FoV 视角范围内的高清 Tile 流，如此一来，终端不再需要解码全部视角的高清视频流，可有效降低终端解码压力。

# 4 项目计划

## 4.1 项目开发计划表

| 序号 | 阶段名称 | 工作事项 | 时间 | 备注（实施前置条件） |
|---|---|---|---|---|
| 1 | 需求阶段 | 设计架构确认 | 第 1 周-第 4 周 | 需要客户和供应商一起讨论整体构架设计确认。 |
| | | 功能需求确认 | | 需要客户方进行功能需求确认，只有功能需求确认后方可后续实施 |
| | | 测试用例大纲确认 | | 供应商按照客户的需求定义测试大纲，并让客户确认 |
| | | 测试平台及测试流确认 | | 需要客户确认测试平台硬件以及测试流媒体的覆盖性 |
| 2 | 设计阶段 | 演示 demo 确认 | 第 5 周-第 15 周 | 基本功能确认工作，不包括特殊功能和性能优化 |

| | | | | |
|---|---|---|---|---|
| | | 总体功能设计 | | 总体功能流程的确认工作 |
| | | 测试用例设计确认 | | 供应商按照客户的需求设计完成测试用例，需要客户进行完整性确认 |
| | | API 接口设计确认 | | 根据设计框架来定义 API 接口并需要客户确认 |
| 3 | 开发阶段 | 功能开发 | 第 16 周-第 40 周 | 开发阶段 |
| | | 接口开发 | | |
| 4 | 测试阶段 | 一轮测试、bug 修改 | 第 41 周--第 50 周 | 项目整体测试阶段、分三轮进行 |
| | | bug 回归，二轮测试 | | |
| | | bug 回归，三轮测试 | | |
| 5 | | 部署上线测试（灰度测试） | 第 51 周--第 52 周 | |

| | | | | |
|---|---|---|---|---|
| | 第一阶段试上线阶段 | 试运行 | | 上线后进行灰度测试，排除主要问题后进行试运行 |